

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
20 November 2003 (20.11.2003)

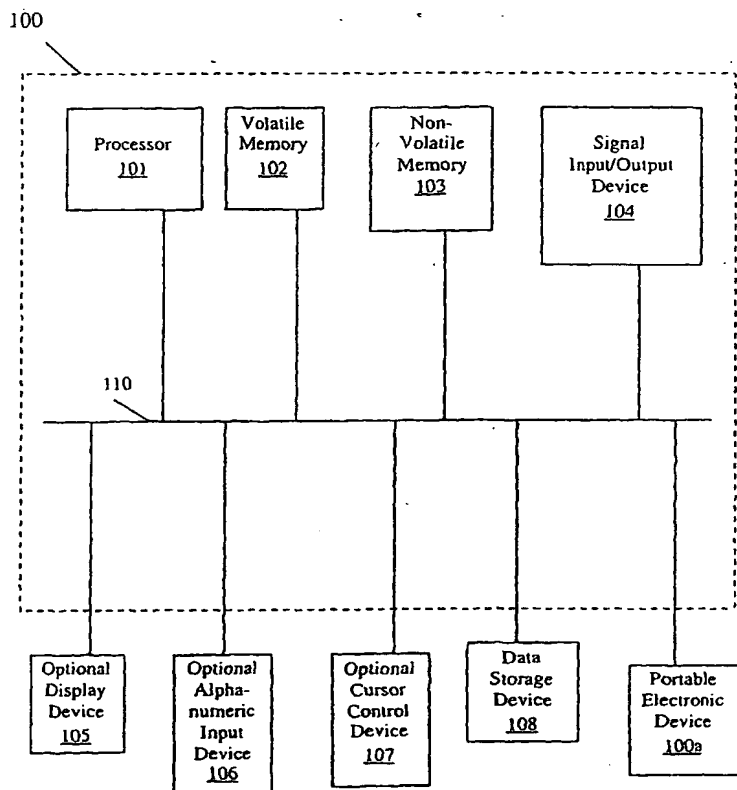
PCT

(10) International Publication Number
WO 03/096340 A2

- (51) International Patent Classification⁷: **G11B 20/00** 10/430,843 5 May 2003 (05.05.2003) US
10/430,477 5 May 2003 (05.05.2003) US
- (21) International Application Number: PCT/US03/14878
- (22) International Filing Date: 10 May 2003 (10.05.2003)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
- | | | |
|------------|-------------------------------|----|
| 60/379,979 | 10 May 2002 (10.05.2002) | US |
| 60/378,011 | 10 May 2002 (10.05.2002) | US |
| 10/218,241 | 13 August 2002 (13.08.2002) | US |
| 10/235,293 | 4 September 2002 (04.09.2002) | US |
| 10/304,390 | 25 November 2002 (25.11.2002) | US |
| 10/325,243 | 18 December 2002 (18.12.2002) | US |
| 10/364,643 | 10 February 2003 (10.02.2003) | US |
| 60/451,231 | 28 February 2003 (28.02.2003) | US |
- (71) Applicants and
(72) Inventors: **RISAN, Hank** [US/US]; 515 Washington Street, Santa Cruz, CA 95060 (US). **FITZGERALD, Edward, Vincent** [US/US]; 100 Peach Terrace, Santa Cruz, CA 95060 (US).
- (74) Agents: **GALLENSON, Mavis, S. et al.**; Ladas & Parry, 5670 Wilshire Boulevard, Suite 2100, Los Angeles, CA 90036 (US).
- (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR MEDIA



(57) Abstract: Method and system for media are described. Particularly, embodiments of a system and method for controlling unauthorized reproduction of protected media on a media storage device are described. Embodiments of a method and system for providing user selectable and location-obscured streaming media are also described. Embodiments of a system and method for providing global media content delivery are also described. Embodiments of a method and system for controlling interaction of deliverable electronic media are also described. Embodiments of a method for controlling recording of media are described. Embodiments of a method and system for controlling reproduction of protected media on a media storage device are also described. Embodiments of a method and system for controlling access of media on a media storage device are also described. Embodiments of a method and system for controlling presentation of media on a media storage device are also described.

Best Available Copy



SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ,
VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND SYSTEM FOR MEDIA

FIELD:

Section 0:

The present invention relates to electronic media. More particularly, the present invention relates to preventing unauthorized recording of protected electronic media disposed on a media storage device.

Sections 1 and 2:

The invention involves a method and system to providing user selectable and location-obscured streaming media.

Section 3:

The present invention generally relates to the field of media. More particularly, the present invention relates to the field of media delivery.

Section 4:

The present invention generally relates to the field of media. More particularly, the present invention relates to the field of media delivery.

Section 5:

The present invention relates to electronic media. More particularly, the present invention relates to restricting interaction of delivered electronic media.

Section 6:

The present invention relates to the recording of electronic media. More particularly, the present invention relates to preventing unauthorized recording of electronic media.

Section 7:

The present invention relates to electronic media. More particularly, the present invention relates to preventing unauthorized recording of electronic media.

Section 8:

The present invention relates to electronic media. More particularly, the present invention relates to preventing unauthorized recording of electronic media disposed on a media storage device.

Section 9:

The present invention relates to electronic media. More particularly, the present invention relates to preventing unauthorized reproduction and/or distribution of demonstration and/or pre-release media disposed on a media storage device.

BACKGROUND:**Section 0:**

One of the problems faced in attempting to effectively control media files on a media storage device, e.g., a CD, in a secure and controlled manner is that current media storage devices need to be compatible with both home media storage device players, e.g., a CD, DVD or other player, and media storage device drives which may be coupled with a computer system. These players/drives were designed in 1982, and the media storage device needs to be backwards compatible with those players/drives.

Media storage device drives are essentially data transducers, meaning they convert bit stream data into electronic waveform that is output, e.g., as an analog waveform, harmonic waveform, which is output to speakers and/or other devices to render sound.

A computer system is more problematic because in addition to its transducing abilities, it may also be: A) a morphogenic system, meaning that a user can take data, reorganize the data, and morph it into other forms on the computer; and/or B) a replicator system, meaning that it can also copy or capture or store or reproduce the data.

The data format of a media storage device, e.g., a compact disk, was designed in 1982 and it was not designed with any security in mind. This is because it was designed to be effective media for data transduction, and as such, did not include provisions to provide for effective copyright protection or Digital Rights Management (DRM).

5

Some companies that have attempted to provide copyright protection are doing so in a way that is designed to exploit inefficiencies or discrepancies between the home player and the media storage device drive coupled with a computer system. To provide media files for both players/drives, those companies do multisession tracks. The media storage device, e.g.,
10 CD/DVD, delivers two sets of data. In one example, a plus sign may be used to indicate that the CD/DVD is a mixed disc, having both data for the computer and music for the home machine. Double clicking on the icon initiates autoplay of the CD/DVD, which in one example, activates a player provided by the CD/DVD.

15

One set of streaming data is for the drive coupled with a computer system (generally called by a proprietary player and delivered in a highly compressed bit form to the computer/user) that may have some kind of digital rights management. For example, when a media storage device is inserted into a device drive coupled with a computer system, the user may be presented with a proprietary player having a bit rate of approximately 128Kbps, which
20 can present a highly compressed version of the original to the user so they will be able to experience the media file.

Disadvantageously, a data stream of 128Kbps is severely degraded from the original. In many instances, common compression ratios of original waveforms are approximately a ten to
25 one compression ratio. A ten to one compression ratio is readily audible. Thus, the user would be experiencing poor quality sound.

The other set of data is an audio file that is accessed by a home music or video system and the user is able to experience the media file. Inserting the media storage device into the home audio/video device enables the user to experience the media file in an uncompressed high quality manner, replicating the original form of the media file.

5

In many instances, all that is needed is a click of the mouse to strip the DRM protection off the media storage device, and the media file becomes available for reproduction and distribution. Alternative means to defeat copyright protection of media files can be as easy as using a magic marker technique. In this technique, a user marks the outer track on a media storage device, e.g., a CD, with a permanent marker, e.g., a Sharpie. When the computer tries to read the first track, it fails, and by default, then reads the next track, usually where the music begins.

Additionally, the media file copyright holders are being sold on the premise that a degraded media file is better than the original because you can't control the original on the computer. Therefore, users may be less likely to use a computer to record/capture/reproduce a poor quality version. Once the user does capture the media file, it is a mediocre sounding copy. This fundamental concept of recording companies giving a less than ideal data version on the CD is in the hope that the lack of sound quality will deter users from recording, copying, etc., the media files.

20

Alternative methods to provide protect and DRM include the use of time clock inefficiencies. For example, one method is to indicate to the computer system that a media file begins further back than where it actually does, which can introduce a series of numerous errors.

25

Home machines, e.g., CD/DVD players coupled with stereos, in comparison to CD/DVD drives coupled with a computer system, are extremely tolerant of errors. Home CD/DVD players

are designed to read from CDs and DVDs that have been mistreated, e.g., scratched, left out of their jewel case, etc. The home players have substantial error correcting capabilities. Thus, if a CD/DVD has data that was given a negative start time, the home players detects that there is no such thing as a negative start time, and then the home player commences playback.

5

However, computers are more gullible, meaning that they believe what you tell them. So if the CD/DVD indicates a negative start time, then the media storage device drive coupled with the computer system may not be able to play a particular media file.

10

There are also legacy issues and compatibility issues. The consumer is being given a faulty product. In many instances, a disclaimer commonly found on current CDs and DVDs says that if the CD/DVD does not function, return the CD/DVD for exchange. Many users may find this intermittent functionality unacceptable and having to return CDDVD may cause the user to postpone or, more severely, cancel future CD/DVD purchases.

15

Applications are readily available via the Internet for the express purpose of producing and exact audio copy of media files on a media storage device. One example is Exact Audio Copy, a freeware software program freely available on the Internet which produces an exact audio copy in .wav file format. Using Exact Audio Copy, circumventing existing protection was accomplished without modification to the existing technology. The Exact Audio Copy application bypasses the multi session data tracks and goes directly to the audio tracks. This was accomplished by loading the Exact Audio Copy onto a computer, inserting a CD, and pressing a button or two to copy the audio tracks.

20

25 Additionally, there are ripping applications, readily available via the Internet, that read the redbook, which enables the ripping application to access the table of contents, and the ripping application goes to the audio tracks where it can rip the audio or video file.

Further, DRM protection methods implemented as a stand alone device, meaning that the DRM and copy protection resides in software that resides on the disk are also problematic. This is because when circumvention of the DRM on the media storage device occurs, little if anything can be done because the DRM controls are also bypassed. There is no communication with the computer or the Internet.

Software DRM solutions are additionally problematic for CDs and DVDs because they frequently do not provide DRM compliance, and it is foreseen that software solutions will not provide DRM protection in the future, particularly with the introduction of new OS's and new media formats. These types of software DRM solutions are difficult to morph into a secure format once operating systems change, a non-morphogenic format on a morphogenic computer system.

In many instances, demo media files are being copied and released prior to release of the actual media file. In other instances, unauthorized copies of protected media files, e.g., CDs and/or DVD's are being released before the release of the movies. In some instances, unauthorized copies of protected media files are outselling legally produced media files.

Further, many of the media player/recorder applications are designed to capture and record incoming media files in a manner that circumvents controls implemented by a media player application inherent to an operating system, e.g., QuickTime for Apple, MediaPlayer for Windows™, etc., or one downloadable from the Internet, e.g., RealPlayer, LiquidAudio, or those provided by webcasters, e.g., PressPlay, for controlling unauthorized recording of media files. Also, many digital recording devices, e.g., mini-disc recorders, MP3 recorders, and the like, can be coupled to a digital output of a computer system, e.g., a USB port, a S/Pdif out, and the like, to capture the media file.

It is also desired to prevent recording applications, such as Total Recorder, Sound Forge, and numerous others, that are adapted to establish a connection with a kernel level driver operable within an operating system to capture and redirect the media file to create an unauthorized reproduction of the media file. It is also desired to prevent recording applications from accessing a kernel-mode media device driver and making unauthorized copies of copyrighted material through some available network, e.g., wireline, wireless, P2P, etc., or through a communicative coupling. It is further desirable to prevent access to a kernel based media device driver by a recording application for the purpose of making unauthorized copies of media files from or to alternative sources, e.g., CD players, DVD players, removable hard drives, personal electronic and/or recording devices, e.g., MP3 recorders, and the like.

Current methods of preventing unauthorized reproduction of protected medial files on media storage device are inadequate.

Section 1:

A flow chart of steps for a conventional Streaming Media method is shown in Figure 1-1.

Section 2:

Figure 2-1 shows a flow chart of steps performed in a conventional Streaming Media method.

Section 3:

Computers have become integral tools used in a wide variety of different applications, such as in finance and commercial transactions, computer-aided design and manufacturing, health care, telecommunication, education, etc. Computers are finding new applications as a result of advances in hardware technology and rapid development in software technology. Furthermore, the functionality of a computer system is dramatically enhanced by coupling these type of stand-alone devices together in order to form a networking environment. Within a networking environment, users may readily exchange files, share information stored on a common database, pool resources, and communicate via electronic mail (e-mail) and video conferencing. Furthermore, computers which are coupled to a networking environment like the Internet provide their users access to data and information from all over the world. Computer systems have become useful in many aspects of everyday life including providing entertainment to their users.

For example, a computer system may be coupled to the Internet and receive music from a wide variety of web site sources. There are different ways that a computer system may receive music via the Internet. One common way is for the computer to receive streaming music from a web site wherein the computer user typically does not choose the songs that he or she listens to but instead receives broadcast music. However, it should be appreciated that there are disadvantages associated with this streaming media technique for receiving music.

One of the disadvantages is that the computer user is unable to choose

specific songs to listen to and have them delivered to his or her computer “on-demand.” Another disadvantage with the streaming music delivery technique is that as more and more computers receive the simultaneously broadcast music, the more and more congested the communication networks of the Internet become. In order to lessen the severity of this type of network congestion, media streamers usually decrease the data rate of the streaming audio in order to be able to handle more clients. However, by decreasing the data rate of the music its fidelity is also degraded.

Some streaming music delivery systems require a participating computer system to acquire a proprietary audio player in order to receive and play music which has been encrypted so that the music is not distributed to others in an uncontrolled fashion. Nevertheless, there are disadvantages associated with this technique also. For example, one of the disadvantages is that the fidelity of the received music can be degraded as a consequence of its encryption and decryption. Additionally, an encryption process typically involves more data being included with the streaming music which may provide additional congestion to communication networks of the Internet. Another disadvantage associated with some proprietary audio players is that they are not compatible with certain computer operating system platforms which consequently restricts the number of computer systems that may utilize them.

Another technique for acquiring music with a computer is by accessing a music file sharing web site via the Internet. Currently there are a variety of different

Internet music file sharing web sites accessible to computers. Typically, these file sharing web sites enable computer users to share music files in a point-to-point manner thereby enabling them to receive music which may then be stored by their computer for use or further distribution. It should be appreciated that there are disadvantages associated with this type of music file sharing technique.

One of the disadvantages is that the music file sharing computer users usually do not compensate the owner (e.g., record company, artist, etc.) of copyrighted music after receiving a copy of it. One of the reasons for this situation is that music file sharing web sites commonly do not control or even monitor the music being exchanged among different computers. And even when these web sites try to record and/or control the distribution of music, it may be thwarted by simply changing the file name of the song thereby enabling it to continue to be distributed freely.

Another disadvantage associated with the music file sharing technique previously described is that the music received may be a low fidelity product. There are different causes for this low fidelity. For example, there is usually no established quality level that has to be met in order to share music files via the Internet. As such, the original "source" of a shared music file is typically music copied from a compact disc (CD) to a computer which may be done in a substandard manner because an operator is not concerned with quality and/or does not even have the tools necessary to produce a high fidelity product. A further disadvantage associated with this music file sharing technique is that as more and more file sharing or swapping occurs, communication network systems of the Internet can get

increasingly more congested thereby reducing the efficiency of data transfer between different computers.

Section 4:

Computers have become integral tools used in a wide variety of different applications, such as in finance and commercial transactions, computer-aided design and manufacturing, health care, telecommunication, education, etc. Computers are finding new applications as a result of advances in hardware technology and rapid development in software technology. Furthermore, the functionality of a computer system is dramatically enhanced by coupling these type of stand-alone devices together in order to form a networking environment. Within a networking environment, users may readily exchange files, share information stored on a common database, pool resources, and communicate via electronic mail (e-mail) and video teleconferencing. Furthermore, computers which are coupled to a networking environment like the Internet provide their users access to data and information from all over the world. Computer systems have become useful in many aspects of everyday life including providing entertainment to their users.

For example, a computer system may be coupled to the Internet and receive music from a wide variety of web site sources. There are different ways that a computer system may receive music via the Internet. One common way is for the computer to receive streaming music from a web site wherein the computer user typically does not choose the songs that he or she listens to but instead receives broadcast music. However, it should be appreciated that there are disadvantages.

associated with this streaming media technique for receiving music.

One of the disadvantages is that the computer user is unable to choose specific songs to listen to and have them delivered to his or her computer "on-demand." Another disadvantage with the streaming music delivery technique is that as more and more computers receive the simultaneously broadcast music, the more and more congested the communication networks of the Internet become. In order to lessen the severity of this type of network congestion, media streamers usually decrease the data rate of the streaming audio in order to be able to handle more clients. However, by decreasing the data rate of the music, its fidelity is also degraded.

Some streaming music delivery systems require a participating computer system to acquire a proprietary audio player in order to receive and play music which has been encrypted so that the music is not distributed to others in an uncontrolled fashion. Nevertheless, there are disadvantages associated with this technique also. For example, one of the disadvantages is that the fidelity of the received music can be degraded as a consequence of its encryption and decryption. Additionally, an encryption process typically involves more data being included with the streaming music which may provide additional congestion to communication networks of the Internet. Another disadvantage associated with some proprietary audio players is that they are not compatible with certain computer operating system platforms which consequently restricts the number of computer systems that may utilize them.

Another technique for acquiring music with a computer is by accessing a music file sharing web site via the Internet. Currently there are a variety of different Internet music file sharing web sites accessible to computers. Typically, these file sharing web sites enable computer users to share music files in a point-to-point manner thereby enabling them to receive music which may then be stored by their computer for use or further distribution. It should be appreciated that there are disadvantages associated with this type of music file sharing technique.

One of the disadvantages is that the music file sharing computer users usually do not compensate the owner (e.g., record company, artist, etc.) of copyrighted music after receiving a copy of it. One of the reasons for this situation is that music file sharing web sites commonly do not control or even monitor the music being exchanged among different computers. And even when these web sites try to record and/or control the distribution of music, it may be thwarted by simply changing the file name of the song thereby enabling it to continue to be distributed freely.

Another disadvantage associated with the music file sharing technique previously described is that the music received may be a low fidelity product. There are different causes for this low fidelity. For example, there is usually no established quality level that has to be met in order to share music files via the Internet. As such, the original "source" of a shared music file is typically music copied from a compact disc (CD) to a computer which may be done in a substandard manner because an operator is not concerned with quality and/or does not even have the

tools necessary to produce a high fidelity product. A further disadvantage associated with this music file sharing technique is that as more and more file sharing or swapping occurs, communication network systems of the Internet can get increasingly more congested thereby reducing the efficiency of data transfer between different computers.

Section 5:

With advancements in hardware and software technology, computers are integral tools utilized in various applications, such as finance, CAD (computer aided design), manufacturing, health care, telecommunication, education, etc. Further, an enhancement in computer functionality can be realized by communicatively coupling computers together to form a network. Within a network environment, computer systems enable users to exchange files, share information stored in common databases, combine or pool resources, communicate via electronic mail (e-mail), and access information on the Internet. Additionally, computers connected to a network environment, e.g., the Internet, provide their users access to data and information from all over the world.

Some of the various types of data that a user can access and share include, but are not limited to, text data such as that found in a word document, graphical data such as that found in pictures, e.g., JPEGs, GIFs, TIFFs, audio data such as that found in music files, e.g., MP3 files, and video data such as that found in moving pictures files, e.g., MPEG, MOV, and AVI files, to name a few. In fact, nearly any type of data can be stored and shared with other computer systems. In many

instances, the material contained within the various data types is copyrighted material.

There are many different types of network environments that can be implemented to facilitate sharing of data between computer systems. Some of the various network environment types include Ethernet, client-server, and wired and/or wireless network environments. A common utilization of a network environment type is for file sharing, such as in a P2P network or point-to-point network. Most P2P networks rely on business models based upon the transfer and redistribution of copyrighted material, e.g., audio files, between computers coupled to a network, e.g., the Internet. A P2P network allows a user to acquire the copyrighted material from a computer, a web site source, or a music broadcaster, and store and share the material with other users throughout the network, in some instances acting as a web site source or a music broadcaster.

It is also common for users sharing files in an uncontrolled manner to use freely distributed or commercially available media player applications to experience, e.g., listen, view, and/or watch, the shared files. In many instances, these media player applications also provide for downloading the media file from a P2P network or from licensed web broadcasters, saving it locally, and then upload the media file onto an unlawful P2P or similar network and/or consumer recording devices. Unlawfully saving a media file can be as simple as selecting the save or record function on a media player application.

Additionally, many of the computers, web sites, and web broadcasters that share copyrighted material commonly do not control or monitor the files being exchanged between computers. Additionally, when web sites attempt to control or restrict the distribution of copyrighted material, e.g., audio files, users seeking to circumvent controls or restrictions can, in many cases, simply utilize the recording functionality of a media player application and save the copyrighted material, rename the particular audio file, and upload the renamed file, rendering attempts to control or restrict its distribution moot.

A disadvantage to the uncontrolled sharing of files, more particularly the downloading, saving, and uploading of copyrighted material, e.g., music files, is that there is currently no effective means to provide compensation to the owner (e.g., record company, lyricist, musician, etc.) of the copyrighted material. Studies have revenue losses in the billions due to unauthorized copying and inaccurate reporting of royalties.

Current methods of sharing music files do not provide adequate file distribution controls or proper accountability with regard to licensing agreements and/or copyright restrictions associated with shared copyrighted material.

Section 6:

With advancements in hardware and software technology, computers are integral tools utilized in various applications, such as finance, CAD (computer aided design), manufacturing, health care, telecommunication, education, etc. Further, an

enhancement in computer functionality can be realized by communicatively coupling computers together to form a network. Within a network environment, computer systems enable users to exchange files, share information stored in common databases, combine or pool resources, communicate via electronic mail (e-mail), and access information on the Internet. Additionally, computers connected to a network environment, e.g., the Internet, provide their users access to data and information from all over the world.

Some of the various types of data that a user can access and share include, but are not limited to, text data such as that found in a word document, graphical data such as that found in pictures, e.g., JPEGs, GIFs, TIFFs, audio data such as that found in music files, e.g., MP3 files, and video data such as that found in moving pictures files, e.g., MPEG, MOV, and AVI files, to name a few. In fact, nearly any type of data can be stored and shared with other computer systems. In many instances, the material contained within the various data types is copyrighted material.

There are many different types of network environments that can be implemented to facilitate sharing of data between computer systems. Some of the various network environment types include Ethernet, client-server, and wired and/or wireless network environments. A common utilization of a network environment type is for file sharing, such as in a P2P network or point-to-point network. Most P2P networks rely on business models based upon the transfer and redistribution of copyrighted material, e.g., audio files, between computers coupled to a network,

e.g., the Internet. A P2P network allows a user to acquire the copyrighted material from a computer, a web site source, or a music broadcaster, and store and share the material with other users throughout the network, in some instances acting as a web site source or a music broadcaster.

It is also common for users sharing media files in an uncontrolled manner to use freely distributed or commercially available media player applications to experience, e.g., listen, view, and/or watch, the shared files. In many instances, these media player applications also provide for downloading the media file from a P2P network or from licensed web broadcasters, saving it locally, and then upload the media file onto an unlawful P2P or similar network and/or consumer recording devices. Unlawfully saving/recording a media file can be as simple as selecting the save or record function on a media player application.

Additionally, many of the computers, web sites, and web broadcasters that share copyrighted material commonly do not control or monitor the files being exchanged between computers. Additionally, when web sites attempt to control or restrict the distribution of copyrighted material, e.g., audio files, users seeking to circumvent controls or restrictions can, in many cases, simply utilize the recording functionality of a media player application and save the copyrighted material, rename the particular audio file, and upload the renamed file, rendering attempts to control or restrict its distribution moot.

Further, many of the media player/recorder applications are designed to

capture and record incoming media files in a manner that circumvents controls implemented by a media player application inherent to an operating system, e.g., QuickTime for Apple, MediaPlayer for Windows™, etc., or one downloadable from the Internet, e.g., RealPlayer, LiquidAudio, or those provided by webcasters, e.g., PressPlay, for controlling unauthorized recording of media files. Additionally, many digital recording devices, e.g., mini-disc recorders, MP3 recorders, and the like, can be coupled to a digital output of a computer system to capture the media file.

It is desired to prevent persons from making unauthorized copies of copyrighted material through some available network, e.g., wireline, wireless, P2P, etc., or through a communicative coupling. It is further desirable to prevent persons from making unauthorized copies of media files from or to alternative sources, e.g., CD players, DVD players, removable hard drives, personal electronic and/or recording devices, e.g., MP3 recorders, and the like.

Current methods of sharing media files do not provide adequate protection against unauthorized recording of the media files.

Section 7:

With advancements in hardware and software technology, computers are integral tools utilized in various applications, such as finance, CAD (computer aided design), manufacturing, health care, telecommunication, education, etc. Further, an enhancement in computer functionality can be realized by communicatively coupling computers together to form a network. Within a network environment, computer

systems enable users to exchange files, share information stored in common databases, combine or pool resources, communicate via electronic mail (e-mail), and access information on the Internet. Additionally, computers connected to a network environment, e.g., the Internet, provide their users access to data and information from all over the world.

Some of the various types of data that a user can access and share include, but are not limited to, text data such as that found in a word document, graphical data such as that found in pictures, e.g., JPEGs, GIFs, TIFFs, audio data such as that found in music files, e.g., MP3 files, and video data such as that found in moving pictures files, e.g., MPEG, MOV, and AVI files, to name a few. In fact, nearly any type of data can be stored and shared with other computer systems. In many instances, the material contained within the various data types is copyrighted material.

There are many different types of network environments that can be implemented to facilitate sharing of data between computer systems. Some of the various network environment types include Ethernet, client-server, and wired and/or wireless network environments. A common utilization of a network environment type is for file sharing, such as in a P2P network or point-to-point network. Most P2P networks rely on business models based upon the transfer and redistribution of copyrighted material, e.g., audio files, between computers coupled to a network, e.g., the Internet. A P2P network allows a user to acquire the copyrighted material from a computer, a web site source, or a music broadcaster, and store and share the

material with other users throughout the network, in some instances acting as a web site source or a music broadcaster.

It is also common for users sharing media files in an uncontrolled manner to use freely distributed or commercially available media player applications to experience, e.g., listen, view, and/or watch, the shared files. In many instances, these media player applications also provide for downloading the media file from a P2P network or from licensed web broadcasters, saving it locally, and then upload the media file onto an unlawful P2P or similar network and/or consumer recording devices. Unlawfully saving/recording a media file can be as simple as selecting the save or record function on a media player application.

Additionally, many of the computers, web sites, and web broadcasters that share copyrighted material commonly do not control or monitor the files being exchanged between computers. Additionally, when web sites attempt to control or restrict the distribution of copyrighted material, e.g., audio files, users seeking to circumvent controls or restrictions can, in many cases, simply utilize the recording functionality of a media player application and save the copyrighted material, rename the particular audio file, and upload the renamed file, rendering attempts to control or restrict its distribution moot.

Further, many of the media player/recorder applications are designed to capture and record incoming media files in a manner that circumvents controls implemented by a media player application inherent to an operating system, e.g.,

QuickTime for Apple, MediaPlayer for Windows™, etc., or one downloadable from the Internet, e.g., RealPlayer, LiquidAudio, or those provided by webcasters, e.g., PressPlay, for controlling unauthorized recording of media files. Additionally, many recording applications can be adapted to establish a connection with the kernel-mode media device driver operable within an operating system, so as to capture and redirect the media file to create an unauthorized recording. Also, many digital recording devices, e.g., mini-disc recorders, MP3 recorders, and the like, can be coupled to a digital output of a computer system to capture the media file.

It is desired to prevent recording applications from accessing a kernel-mode media device driver and making unauthorized copies of copyrighted material through some available network, e.g., wireline, wireless, P2P, etc., or through a communicative coupling. It is further desirable to prevent access to a kernel-based media device driver by a recording application for the purpose of making unauthorized copies of media files from or to alternative sources, e.g., CD players, DVD players, removable hard drives, personal electronic and/or recording devices, e.g., MP3 recorders, and the like.

Current methods of sharing media files do not provide adequate protection against unauthorized recording of the media files.

Section 8:

One of the problems faced in attempting to effectively control media files on a media storage device, e.g., a compact disk (CD), in a secure and controlled manner

is that current media storage devices need to be compatible with both home media storage device players, e.g., a CD, digital versatile disk (DVD) or other player, and media storage device drives which may be connected to a computer system. Many of these players/drives were designed in 1982, and the media storage device needs to be backwards compatible with those players/drives.

Media storage device drives are essentially data transducers, meaning they convert bit stream data into electronic waveform that is output, e.g., as an analog waveform, harmonic waveform, to speakers and/or other devices to render sound.

A computer system is more problematic because in addition to its transducing abilities, it may also be: A) a morphogenic system, meaning that a user can take data, reorganize the data, and morph it into other forms on the computer; and/or B) a replicator system, meaning that it can also copy or capture or store or reproduce the data.

The data format of a media storage device, e.g., a CD, was designed in 1982 and it was not designed with any security in mind. This is because it was designed to be effective media for data transduction, and as such, did not include provisions for effective copyright protection or Digital Rights Management (DRM).

Some companies that have attempted to provide copyright protection are doing so in a way that is designed to exploit inefficiencies or discrepancies between the home player and the media storage device drive connected to a computer

system. To provide media files for both players/drives, those companies do multisession tracks. The media storage device, e.g., CD/DVD, delivers two sets of data. In one example, a plus sign may be used to indicate that the CD/DVD is a mixed disc, having both data for the computer and music for the home machine. Double clicking on the icon initiates autoplay of the CD/DVD, which in one example, activates a player provided by the CD/DVD.

One set of streaming data is for the media storage device drive connected to a computer system (generally requested by a proprietary player and delivered in a highly compressed bit form to the computer/user) that may have some kind of digital rights management. For example, when a media storage device is inserted into a device drive connected to a computer system, the user may be presented with a proprietary player having a bit rate of approximately 128Kbps, which can present a highly compressed version of the original to the user so they will be able to experience the media file.

Disadvantageously, a data stream of 128Kbps is severely degraded from the original media. In many instances, common compression ratios of original waveforms are approximately a ten to one compression ratio. A ten to one compression ratio typically results in degradation that is readily audible. Thus, the user would be experiencing poor quality sound.

The other set of data stored by a CD/DVD is an audio file that is accessed by a home music or video system and the user is able to experience the media file.

Inserting the media storage device into the home audio/video device enables the user to experience the media file in an uncompressed high quality manner, replicating the original form of the media file.

In many instances, all that is needed is a click of the mouse to strip the DRM protection off the media storage device, and the media file becomes available for reproduction and distribution. Alternative means to defeat copyright protection of media files can be as easy as using a magic marker technique. In this technique, a user marks the outer track on a media storage device, e.g., a CD, with a permanent marker, e.g., a Sharpie. When the computer tries to read the first track, it fails, and by default, then reads the next track, usually where the music begins.

Additionally, the media file copyright holders are being sold on the premise that a degraded media file is better than the original because you can't control the original on the computer. Therefore, users may be less likely to use a computer to record/capture/reproduce a poor quality version. Once the user does capture the media file, it is a mediocre sounding copy. This fundamental concept of recording companies giving a less than ideal data version on the CD is in the hope that the lack of sound quality will deter users from recording, copying, etc., the media files.

Alternative methods to provide protection and DRM include the use of time clock inefficiencies. For example, one method is to indicate to the computer system that a media file begins further back than where it actually does, which can introduce a series of numerous errors.

Home machines, e.g., CD/DVD players coupled with stereos, in comparison to CD/DVD drives coupled with a computer system, are extremely tolerant of errors. Home CD/DVD players are designed to read from CDs and DVDs that have been mistreated, e.g., scratched, left out of their jewel case, etc. The home players have substantial error correcting capabilities. Thus, if a CD/DVD has data that was given a negative start time, the home players detects that there is no such thing as a negative start time, and then the home player commences playback.

However, computers are more "gullible," meaning that they believe what you tell them. So if the CD/DVD indicates a negative start time, then the media storage device drive connected to a the computer system may not be able to play a particular media file.

There are also legacy issues and compatibility issues. The consumer is being given a faulty product. In many instances, a disclaimer commonly found on current CDs and DVDs says that if the CD/DVD does not function, return the CD/DVD for exchange. Many users may find this intermittent functionality unacceptable and having to return CDDVD may cause the user to postpone or, more severely, cancel future CD/DVD purchases.

Applications are readily available via the Internet for the express purpose of producing an exact audio copy of media files on a media storage device. One example is Exact Audio Copy, a freeware software program freely available on the Internet which produces an exact audio copy in .wav file format. Using Exact Audio

Copy, circumventing existing protection can be accomplished without modification to the existing technology. The Exact Audio Copy application bypasses the multisession data tracks and goes directly to the audio tracks. This can be accomplished by loading the Exact Audio Copy onto a computer, inserting a CD, and pressing a button or two to copy the audio tracks.

Additionally, there are "ripping" applications, readily available via the Internet, that read the redbook, which enables the ripping application to access the table of contents, and the ripping application goes to the audio tracks where it can "rip" the audio or video file.

Further, DRM protection methods implemented as a stand alone device, meaning that the DRM and copy protection resides in software that resides on the disk are also problematic. This is because when circumvention of the DRM on the media storage device occurs, little if anything can be done because the DRM controls are also bypassed. There may not be any communication with the computer or the Internet.

Software DRM solutions are additionally problematic for CDs and DVDs because they frequently do not provide DRM compliance, and it is foreseen that software solutions will not provide DRM protection in the future, particularly with the introduction of new computer operating systems and new media formats. These types of software DRM solutions are difficult to morph into a secure format once operating systems change.

In many instances, demo media files are being copied and released prior to release of the actual media file. In other instances, unauthorized copies of protected media files, e.g., CDs and/or DVDs are being released before the release of the music and/or the movies. In some instances, unauthorized copies of protected media files are outselling legally produced media files.

Further, many of the media player/recorder applications are designed to capture and record incoming media files in a manner that circumvents controls implemented by a media player application inherent to an operating system, e.g., QuickTime for Apple, MediaPlayer for Windows™, etc., or downloadable from the Internet, e.g., RealPlayer, LiquidAudio, or those provided by webcasters, e.g., PressPlay, for controlling unauthorized recording of media files. Also, many digital recording devices, e.g., mini-disc recorders, MP3 recorders, and the like, can be coupled to a digital output of a computer system, e.g., a USB port, a S/Pdif out, and the like, to capture the media file.

It is also desired to prevent recording applications, such as Total Recorder, Sound Forge, and numerous others, that are adapted to establish a connection with a kernel level driver operable within an operating system to capture and redirect the media file to create an unauthorized reproduction of a media file. It is also desired to prevent recording applications from accessing a kernel-mode media device driver and making unauthorized copies of copyrighted material through some available network, e.g., wireline, wireless, P2P, etc., or through a communicative coupling. It is further desirable to prevent access to a kernel based media device driver by a

recording application for the purpose of making unauthorized copies of media files from or to alternative sources, e.g., CD players, DVD players, removable hard drives, personal electronic and/or recording devices, e.g., MP3 recorders, and the like.

Current methods of preventing unauthorized reproduction of protected medial files on media storage device are inadequate.

Section 9:

One of the problems faced in attempting to effectively control media files on a media storage device, e.g., a compact disk (CD), in a secure and controlled manner is that current media storage devices need to be compatible with both home media storage device players, e.g., a CD, digital versatile disk (DVD) or other player, and media storage device drives which may be connected to a computer system. Many of these players/drives were designed in 1982, and the media storage device needs to be backwards compatible with those players/drives.

Media storage device drives are essentially data transducers, meaning they convert bit stream data into electronic waveform that is output, e.g., as an analog waveform, harmonic waveform, to speakers and/or other devices to render sound.

A computer system is more problematic because in addition to its transducing abilities, it may also be: A) a morphogenic system, meaning that a user can take data, reorganize the data, and morph it into other forms on the computer; and/or B)

a replicator system, meaning that it can also copy or capture or store or reproduce the data.

The data format of a media storage device, e.g., a CD, was designed in 1982 and it was not designed with any security in mind. This is because it was designed to be effective media for data transduction, and as such, did not include provisions for effective copyright protection or Digital Rights Management (DRM).

Some companies that have attempted to provide copyright protection are doing so in a way that is designed to exploit inefficiencies or discrepancies between the home player and the media storage device drive connected to a computer system. To provide media files for both players/drives, those companies do multisession tracks. The media storage device, e.g., CD/DVD, delivers two sets of data. In one example, a plus sign may be used to indicate that the CD/DVD is a mixed disc, having both data for the computer and music for the home machine. Double clicking on the icon initiates autoplay of the CD/DVD, which in one example, activates a player provided by the CD/DVD.

One set of streaming data is for the media storage device drive connected to a computer system (generally requested by a proprietary player and delivered in a highly compressed bit form to the computer/user) that may have some kind of digital rights management. For example, when a media storage device is inserted into a device drive connected to a computer system, the user may be presented with a proprietary player having a bit rate of approximately 128Kbps, which can present a

highly compressed version of the original to the user so they will be able to experience the media file.

Disadvantageously, a data stream of 128Kbps is severely degraded from the original media. In many instances, common compression ratios of original waveforms are approximately a ten to one compression ratio. A ten to one compression ratio typically results in degradation that is readily audible. Thus, the user would be experiencing poor quality sound.

The other set of data stored by a CD/DVD is an audio file that is accessed by a home music or video system and the user is able to experience the media file. Inserting the media storage device into the home audio/video device enables the user to experience the media file in an uncompressed high quality manner, replicating the original form of the media file.

In many instances, all that is needed is a click of the mouse to strip the DRM protection off the media storage device, and the media file becomes available for reproduction and distribution. Alternative means to defeat copyright protection of media files can be as easy as using a magic marker technique. In this technique, a user marks the outer track on a media storage device, e.g., a CD, with a permanent marker, e.g., a Sharpie. When the computer tries to read the first track, it fails, and by default, then reads the next track, usually where the music begins.

Additionally, the media file copyright holders are being sold on the premise

that a degraded media file is better than the original because you can't control the original on the computer. Therefore, users may be less likely to use a computer to record/capture/reproduce a poor quality version. Once the user does capture the media file, it is a mediocre sounding copy. This fundamental concept of recording companies giving a less than ideal data version on the CD is in the hope that the lack of sound quality will deter users from recording, copying, etc., the media files.

Alternative methods to provide protection and DRM include the use of time clock inefficiencies. For example, one method is to indicate to the computer system that a media file begins further back than where it actually does, which can introduce a series of numerous errors.

Home machines, e.g., CD/DVD players coupled with stereos, in comparison to CD/DVD drives coupled with a computer system, are extremely tolerant of errors. Home CD/DVD players are designed to read from CDs and DVDs that have been mistreated, e.g., scratched, left out of their jewel case, etc. The home players have substantial error correcting capabilities. Thus, if a CD/DVD has data that was given a negative start time, the home players detects that there is no such thing as a negative start time, and then the home player commences playback.

However, computers are more "gullible," meaning that they believe what you tell them. So if the CD/DVD indicates a negative start time, then the media storage device drive connected to a the computer system may not be able to play a particular media file.

There are also legacy issues and compatibility issues. The consumer is being given a faulty product. In many instances, a disclaimer commonly found on current CDs and DVDs says that if the CD/DVD does not function, return the CD/DVD for exchange. Many users may find this intermittent functionality unacceptable and having to return CDDVD may cause the user to postpone or, more severely, cancel future CD/DVD purchases.

Applications are readily available via the Internet for the express purpose of producing an exact audio copy of media files on a media storage device. One example is Exact Audio Copy, a freeware software program freely available on the Internet which produces an exact audio copy in .wav file format. Using Exact Audio Copy, circumventing existing protection can be accomplished without modification to the existing technology. The Exact Audio Copy application bypasses the multisession data tracks and goes directly to the audio tracks. This can be accomplished by loading the Exact Audio Copy onto a computer, inserting a CD, and pressing a button or two to copy the audio tracks.

Additionally, there are "ripping" applications, readily available via the Internet, that read the redbook, which enables the ripping application to access the table of contents, and the ripping application goes to the audio tracks where it can "rip" the audio or video file.

Further, DRM protection methods implemented as a stand alone device, meaning that the DRM and copy protection resides in software that resides on the disk are also problematic. This is because when circumvention of the DRM on the

media storage device occurs, little if anything can be done because the DRM controls are also bypassed. There may not be any communication with the computer or the Internet.

Software DRM solutions are additionally problematic for CDs and DVDs because they frequently do not provide DRM compliance, and it is foreseen that software solutions will not provide DRM protection in the future, particularly with the introduction of new computer operating systems and new media formats. These types of software DRM solutions are difficult to morph into a secure format once operating systems change.

In many instances, demo media files are being copied and released prior to release of the actual media file. In other instances, unauthorized copies of protected media files, e.g., CDs and/or DVDs are being released before the release of the music and/or the movies. In some instances, unauthorized copies of protected media files are outselling legally produced media files.

Further, many of the media player/recorder applications are designed to capture and record incoming media files in a manner that circumvents controls implemented by a media player application inherent to an operating system, e.g., QuickTime for Apple, MediaPlayer for Windows™, etc., or downloadable from the Internet, e.g., RealPlayer, LiquidAudio, or those provided by webcasters, e.g., PressPlay, for controlling unauthorized recording of media files. Also, many digital recording devices, e.g., mini-disc recorders, MP3 recorders, and the like, can be

coupled to a digital output of a computer system, e.g., a USB port, a S/Pdif out, and the like, to capture the media file.

It is also desired to prevent recording applications, such as Total Recorder, Sound Forge, and numerous others, that are adapted to establish a connection with a kernel level driver operable within an operating system to capture and redirect the media file to create an unauthorized reproduction of a media file. It is also desired to prevent recording applications from accessing a kernel-mode media device driver and making unauthorized copies of copyrighted material through some available network, e.g., wireline, wireless, P2P, etc., or through a communicative coupling. It is further desirable to prevent access to a kernel based media device driver by a recording application for the purpose of making unauthorized copies of media files from or to alternative sources, e.g., CD players, DVD players, removable hard drives, personal electronic and/or recording devices, e.g., MP3 recorders, and the like.

Current methods of preventing unauthorized reproduction of protected medial files on media storage device are inadequate.

SUMMARY:**Section 0:**

Accordingly, a need exists for a method and system that controls unauthorized reproduction of protected media files disposed on a media storage device. Embodiments of the present invention satisfy the above mentioned needs.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

Section 1:

Embodiments of the present invention allow video streaming to be done without proprietary encryption and/or decryption tools or without a proprietary video player.

Section 2:

Embodiments of the present invention allow video streaming to be done without proprietary encryption and/or decryption tools or without a proprietary video player.

Section 3:

Accordingly, a need exists for a method and system for delivering "on-demand" high fidelity music to computer systems via the Internet which does not involve proprietary audio players and/or encryption of the music. A further need exists for a method and system which meets the above need and does not overly congest the communication networks of the Internet. Another need exists for a method and system which meets the above needs and monitors the music delivered in order to compensate the owner of copyrighted music for it. The present invention provides one or more embodiments which accomplish the above mentioned needs.

Specifically, one embodiment of the present invention enables delivery of "on-demand" high fidelity media content to computers via the Internet while restricting unauthorized users from directly retrieving media content from its source database. Once the computer receives the media, it is stored using hidden directories so that it may not be easily shared with others. Within the present embodiment, there are different functionality that are implemented in order to protect and monitor the media content source. For example, the actual address location of the media database is hidden from content recipients while its address directory is periodically change making past addresses obsolete. Additionally, an access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media content requests. By implementing these and other functionality, the present embodiment restricts redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

In another embodiment, the present invention provides a method for providing media content from a source database to a computer while preventing unauthorized access to the source database. The method includes transmitting via a communication network a first request for a media content list from the computer. Additionally, the method includes transmitting to the computer via the communication network the media content play list together with a unique identification, in response to receiving the first request. The method also includes transmitting a second request for delivery of a media content together with the unique identification, in response to receiving the media content list. Furthermore, the method includes transmitting to the computer via the communication network an access key together with an address for the source database containing the media content, in response to the unique identification of the second request being valid. Moreover, the method includes transmitting a third request together with the access key, in response to receiving the access key together with the address of the source database. The method also includes transmitting the media content to the computer, in response to the access key being valid.

In yet another embodiment, the present invention provides a system for providing a media content stored by a content server to a computer while preventing unauthorized access to the media content stored by the content server. The system includes means for transferring via a communication network a first request for a media content list from the computer. Furthermore, the system includes means for transferring to the computer via the communication network the media content list together with a unique identification, in response to receiving the first request.

Additionally, the system includes means for transferring a second request for delivery of a media content of the media content list together with the unique identification, in response to receiving the media content list. The system also includes means for transferring to the computer via the communication network a time sensitive access key together with an address of the content server containing the media content, in response to the unique identification of the second request being valid. Moreover, the system includes means for transferring a third request together with the time sensitive access key to the content server, in response to receiving the time sensitive access key together with the address of the content server. The system also includes means for transferring the media content to the computer, in response to the time sensitive access key being valid.

In still another embodiment, the present invention provides a computer readable medium having computer readable code embodied therein for causing a system to perform: transmitting via a communication network a first request for a media content list from a computer; transmitting to the computer via the communication network the media content list together with a unique identification, in response to receiving the first request; transmitting a second request for delivery of a media content of the media content list together with the unique identification, in response to receiving the media content list; transmitting to the computer via the communication network an access key together with an address for the media content, in response to the unique identification of the second request being valid; transmitting a third request together with the access key, in response to receiving the access key together with the address of the media content; transmitting the

media content to the computer, in response to the access key being valid.

The present invention provides these advantages and others which will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of embodiments in accordance with the present invention.

Section 4:

Accordingly, a need exists for a method and system for delivering “on-demand” high fidelity music to computer systems via the Internet which does not involve proprietary audio players and/or encryption of the music. A further need exists for a method and system which meets the above need and does not overly congest the communication networks of the Internet. Another need exists for a method and system which meets the above needs and monitors the music delivered in order to compensate the owner of copyrighted music for it. The present invention provides one or more embodiments which accomplish the above mentioned needs.

Specifically, one embodiment of the present invention enables global delivery of “on-demand” high fidelity media content to client computers via a network, such as, the Internet or a wide area network (WAN) while restricting unauthorized users from directly retrieving media content from its sources. The present embodiment includes a global media content delivery network that may include multiple “points of presence” which may be located throughout the world. Each point of presence may store a portion or the entirety of a media content library that may be provided to

client devices. Each one of the points of presence may provide media content to client devices in their respective vicinity of the world. Once a client receives media, it is stored using hidden directories to prevent easy redistribution with other devices. An access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media requests. By implementing these and other functionality, the present embodiment restricts redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

In another embodiment, the present invention provides a system for providing media content while preventing unauthorized access to a plurality of content sources. The system includes a media content library having a plurality of media content pieces. Furthermore, the system includes a global media content delivery network having the plurality of content sources. It is understood that each content source of the plurality of content sources stores a portion of the media content library. Additionally, the system includes a security system coupled to the plurality of content sources for allowing delivery of a media content piece from the media content library to an authorized client device while preventing unauthorized access to the media content library stored by the plurality of content sources.

In yet another embodiment, the present invention provides a method for providing media content to a plurality of client devices from a plurality of content sources while preventing unauthorized access to the media content. The method includes transmitting a first request for a media content list of a media content library from a client device of the plurality of client devices. In response to receiving the

first request, the method includes transmitting to the client device the media content play list together with a unique identification. Furthermore, in response to receiving the media content list, the method includes transmitting a second request for delivery of the media content together with the unique identification from the client device. Provided the unique identification is valid, the method includes transmitting to the client device an access key together with a location of a content source of the plurality of content sources containing the media content. Additionally, in response to receiving the access key together with the location of the content source, the method includes transmitting a third request together with the access key to the content source from the client device. Provided the access key is valid, the method includes transmitting the media content to the client device.

In still another embodiment, the present invention provides a system for providing media content while restricting unauthorized access to a plurality of content sources. The system includes a media content library having a plurality of media content pieces. Additionally, the system includes a global media content delivery network having the plurality of content sources. It is appreciated that each content source of the plurality of content sources stores a portion of the media content library. Furthermore, the system includes an authorization system coupled to the plurality of content sources for regulating delivery of a media content piece from the media content library to an authorized client computer while restricting unauthorized access to the media content library stored by the plurality of content sources.

While particular embodiments of the present invention have been specifically described within this summary, it is noted that the invention is not limited to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents which may be included within the spirit and scope of the invention as herein described by the specification and Claims.

Section 5:

Accordingly, a need exists for a method that provides control of the distribution of media content shared through a network environment, e.g., the Internet. Further, a need exists for a method that provides compliance with copyright restrictions and/or licensing agreements associated with the media content being shared. Embodiments of the present invention satisfy the above mentioned needs.

In one embodiment, the present invention provides a method of controlling interaction of deliverable electronic media that is comprised of detecting a media player application operable within a computer system. The media player application enables the computer system to present contents of a media file. The present method is further comprised of governing within the media player application a function that enables non-compliance with a usage restriction applicable to the media file. The present method is further comprised of controlling the output of the media file. The controlling is performed by a compliance mechanism coupled to the computer system. The compliance mechanism is for enabling compliance with the usage restriction applicable to the media file.

In another embodiment, the present invention provides computer implementable instructions stored on a computer readable medium, the instructions for causing a compliance mechanism to perform a method of controlling client interaction of a media file. The method is comprised of discovering a media player application operable within a client computer system. The media player application is for presenting contents of a media file deliverable to the client computer system. The present method is further comprised of regulating a function of the media player application that does not comply with usage restrictions applicable to the media file. The present method further includes controlling output of the media file. The compliance mechanism performs the controlling and also enables compliance with the usage restriction.

In another embodiment, the present invention provides a method for media file usage restriction compliance comprising means for detecting a media player application operable on a client computer system and for presenting contents of a media file. The present method further comprises means for governing a function of said media player application that does not comply with a usage restriction applicable to a media file. The present method further includes means for controlling output of the media file. A compliance mechanism coupled to the client computer system performs the restricting and also enables compliance with the usage restriction applicable to the media file.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the

following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

Section 6:

Accordingly, a need exists for a method that prevents unauthorized recording of media files. Further, a need exists for a method that selectively prevents unauthorized recording of media files. Embodiments of the present invention satisfy the above mentioned needs.

In one embodiment, a method of preventing unauthorized recording of electronic media is comprised of activating a compliance mechanism in response to receiving media content by a client system. The compliance mechanism is coupled to the client system. The media content presentation application is operable and coupled to the compliance mechanism. The method is further comprised of controlling a data output path of the client computer with the compliance mechanism. The method is further comprised of directing the media content via the data output path to a custom media device for selectively restricting output of the media content. The custom media device is coupled to the compliance mechanism and to the media content presentation application. The method is further comprised of preventing a recording application coupled to the client computer system from recording the media content file when recording violates usage restriction applicable to the media content.

In another embodiment, the present invention provides computer

implementable instructions stored on a computer readable medium, the instructions for causing a client system to perform a method of restricting recording of media content. The present method is comprised of animating a compliance mechanism coupled to the client system. The animating is in response to the client system receiving media content. The client system has a media content presentation application coupled thereto and operable with the compliance mechanism. The present method is further comprised of managing an output path of the client computer with the compliance mechanism. The present method is further comprised of governing said media content to a custom media device via the output path to a custom media device for selectively restricting output of said media content.

In another embodiment, the present invention provides a method for restricting recording of media files comprising means for activating a compliance mechanism to control a data output path of a client system. The activating is in response to said client system receiving media content. The compliance mechanism is coupled to the client system and operable in conjunction with a media content presentation application coupled to the client system and operable thereon. The present method further comprises means for directing the media content to a custom media device via said data output path controlled by said compliance mechanism, for selectively restricting output of said media content.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in

the various drawing figures.

Section 7:

Accordingly, a need exists for a method that prevents unauthorized recording of media files. Further, a need exists for a method that selectively prevents unauthorized recording of media files. Embodiments of the present invention satisfy the above mentioned needs.

In one embodiment, a method of preventing unauthorized recording of electronic media is comprised of activating a compliance mechanism in response to a client system receiving media content. The compliance mechanism is coupled to the client system. The client system has a media content presentation application operable thereon and is coupled to the compliance mechanism. The present method further comprises controlling a data path of a kernel-mode media device driver of the client computer with the compliance mechanism upon detection of a kernel streaming mechanism operable on the client system. The method further comprises directing the media content from the kernel-mode media device driver to a media device driver coupled with the compliance mechanism, via the data path, for selectively restricting output of the media content. The method further comprises preventing the media content from being returned from the kernel-mode media device driver to a recording application coupled to the client system from recording the media content when the recording violates a usage restriction applicable to the media content. The present method further comprises allowing the media content to be returned from the kernel-mode device driver to a recording application coupled to

the client system to record the media content when the recording complies with a usage restriction applicable to the media content.

In another embodiment, the present invention provides computer implementable instructions stored on a computer readable medium, the instructions are for causing a client system to perform a method of restricting recording of media content. The present method is comprised of animating a compliance mechanism coupled to the client system. The animating is in response to the client system receiving media content. The client system has a media content presentation application coupled thereto and is operable with the compliance mechanism. The present method further comprises managing a data path of a kernel-mode media device driver of the client system with the compliance mechanism upon discovery of a kernel streaming mechanism operable on the client system. The present method further comprises governing the media content from the kernel-mode media device driver to a media device driver coupled with the compliance mechanism via the data path, for selectively restricting output of the media content.

In another embodiment, the present invention provides a system for preventing unauthorized recording of electronic media comprising means for activating a compliance mechanism to control a data path of a client system. The activating is in response to the client system receiving media content. The data path is a path of a kernel-mode media device driver in an operating system operable on the client system. The compliance mechanism is coupled to the client system and operable in conjunction with a media content presentation application coupled to the

client system and operable thereon. The system further comprises means for directing the media content from the kernel-mode media device driver to a media device driver via the data path controlled by the compliance mechanism, for selectively restricting output of the media content.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

Section 8:

Accordingly, a need exists for a method and system that controls unauthorized reproduction of protected media files disposed on a media storage device. Embodiments of the present invention satisfy the above mentioned needs.

In one embodiment, a method for preventing unauthorized reproduction of protected media on a media device is described. In one embodiment, the method is comprised of activating an autorun mechanism disposed on said media storage device. The activating is in response to a device drive coupled with a computer system receiving the media storage device. The autorun mechanism for initiating installing a compliance mechanism on the computer system. The method further includes installing the compliance mechanism on the computer system. The compliance mechanism communicatively coupled with the computer system when installed thereon. The compliance mechanism is for enforcing compliance with a

usage restriction applicable to the protected media. The method further includes obtaining control of a data input pathway operable on the computer system. The method further includes preventing the protected media on the media storage device from being captured by an extractor mechanism via the data input pathway while enabling presentation of the protected media.

In another embodiment, a system for preventing unauthorized reproduction of protected media on a media storage device is described. In one embodiment, the system is comprised of a compliance mechanism disposed on the media storage device. The compliance mechanism is configured to be installed on and communicatively coupled with a computer system. The compliance mechanism is for complying with a usage restriction applicable to the protected media. The system further includes an autorun mechanism disposed on the media storage device. The autorun mechanism is configured to install the compliance mechanism and a presentation mechanism on the computer system. The system further includes the compliance mechanism being configured to prevent capturing of the protected media by an extraction mechanism via a data input pathway on the computer system while presenting the protected media.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

Section 9:

Accordingly, a need exists for a method and system that controls presentation of content on a media storage device. Embodiments of the present invention satisfy the above mentioned needs.

In one embodiment, a method for controlling presentation of content on a media storage device is described. In one embodiment, the method is comprised of verifying the presence of a content presentation mechanism and a usage compliance mechanism on a computer system operated by a recipient to whom the media storage device is distributed. The usage compliance mechanism includes a file system filter driver for controlling data reads associated with the content. The present method further includes permitting the recipient to experience the content via the computer system provided the usage compliance mechanism is present on the computer system and the computer system is communicatively coupled with a network and wherein a server in the network authorizes the recipient to experience the content. The present method further includes presenting the content to the recipient via the content presentation mechanism. The content presentation mechanism is communicatively coupled with the usage compliance mechanism. The content presentation mechanism is enabled to present the content provided the content presentation mechanism is communicatively coupled with the server.

In another embodiment, a system for controlling presentation of content on a media storage device is described. In one embodiment, the system is comprised of a detecting means for detecting presence of a usage compliance mechanism

operable on a computer system operated by a recipient to whom the media storage device is distributed. The detecting means is also for detecting presence of a content presentation mechanism operable on the computer system. The usage compliance mechanism including a file system filter driver for controlling data reads associated with the content. The system further includes an authorizing means for authorizing the recipient to experience the content provided the usage compliance mechanism is installed on the computer system and the computer system is communicatively coupled with a network and wherein a server in the network issues authorization allowing the recipient to experience the content. The system further includes a content presenting means for presenting the content to the recipient. The content presentation means is communicatively coupled with the usage compliance mechanism. The content presenting means is enabled to present the content to the recipient provided the content presentation means is communicatively coupled with the server.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS:

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

FIGURE 1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 2 is a block diagram of an exemplary network environment that can be utilized in accordance with an embodiment of the present invention.

FIGURE 3 is a block diagram of various exemplary functional components of a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 5A is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with one embodiment of the present invention.

FIGURE 5B is a data flow block diagram showing an implementation of a component of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with another embodiment of the present invention.

FIGURE 5C is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files, in accordance with one embodiment of the present invention.

5 FIGURE 5D is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files through media file capture at a kernel level, in accordance with one embodiment of the present invention.

10 FIGURE 6A is a block diagram of an environment for preventing unauthorized copying of a media file, in accordance with one embodiment of the present invention.

FIGURES 7A, 7B, and 7C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright compliance mechanism to a network of client and server computer systems.

15

FIGURE 8 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an embodiment of the present invention.

20

FIGURE 9 is a block diagram of components of a copyright compliance mechanism installable from a media storage device upon which protected media files are disposed, in accordance with one embodiment of the present invention.

25

FIGURE 10 is a block diagram of a communicative environment for dynamic updating of a copyright compliance mechanism installed from a media storage device onto a client computer system, in accordance with one embodiment of the present invention.

FIGURE 11 is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized reproduction of a protected media file located on a media storage device, in accordance with one embodiment of the present invention.

5

PRIOR ART FIGURE 1-1 is a flow chart of steps performed in a conventional Prior Art Streaming Media method.

FIGURE 1-2 is a flow chart of steps performed in accordance with one embodiment of the present invention.

FIGURE 1-3 is a flow chart of steps performed in accordance with another embodiment of the present invention.

FIGURE 1-4 is a block diagram of an exemplary method and system for implementing access keys in accordance with an embodiment of the present invention.

FIGURE 1-5 is a block diagram of an exemplary computer system in accordance with an embodiment of the present invention.

The drawings referred to in this description should not be understood as being drawn to scale except if specifically noted.

PRIOR ART FIGURE 2-1 is a flow chart of steps performed in a conventional Prior Art Streaming Media method.

FIGURE 2-2 is a flow chart of steps performed in accordance with one embodiment of the present invention.

FIGURE 2-3 is a flow chart of steps performed in accordance with another embodiment of the present invention.

FIGURE 2-4 is a block diagram of an exemplary method and system for implementing access keys in accordance with an embodiment of the present invention.

FIGURE 2-5 is a block diagram of an exemplary computer system in accordance with an embodiment of the present invention.

FIGURE 2-6 is a diagram of an exemplary global media content delivery system in accordance with an embodiment of the present invention.

Figure 3-1 is a block diagram of an embodiment of an exemplary computer system that may be used in accordance with the present invention.

Figure 3-2 is a block diagram of an exemplary network that may be used in accordance with an embodiment of the present invention.

Figure 3-3 is a block diagram of an exemplary system for implementing access keys in accordance with an embodiment of the present invention.

Figures 3-4A and 3-4B are a flowchart of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device.

Figure 4-1 is a block diagram of an embodiment of an exemplary computer system that may be used in accordance with the present invention.

Figure 4-2 is a block diagram of an exemplary network that may be used in accordance with an embodiment of the present invention.

Figure 4-3 is a block diagram of an exemplary system for implementing access keys in accordance with an embodiment of the present invention.

Figures 4-4A and 4-4B are a flowchart of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device.

Figure 4-5 is a diagram of an exemplary global media content delivery system in accordance with an embodiment of the present invention.

FIGURE 5-1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 5-2 is a block diagram of an exemplary network environment that can be utilized in accordance with an embodiment of the present invention.

FIGURE 5-3 is a block diagram of various exemplary functional components of a

copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 5-4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURES 5-5A, 5-5B, and 5-5C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright compliance mechanism to a network of client and server computer systems.

FIGURE 5-6 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an embodiment of the present invention.

FIGURE 6-1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 6-2 is a block diagram of an exemplary network environment that can be utilized in accordance with an embodiment of the present invention.

FIGURE 6-3 is a block diagram of various exemplary functional components of a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 6-4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 6-5A is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with one embodiment of the present invention.

FIGURE 6-5B is a data flow block diagram showing an implementation of a component of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with another embodiment of the present invention.

FIGURE 6-5C is a data flow block diagram showing an implementation of copyright compliance mechanism for preventing unauthorized output of media files, in accordance with one embodiment of the present invention.

FIGURE 6-6A is a block diagram of an environment for preventing unauthorized copying of a media file, in accordance with one embodiment of the present invention.

FIGURES 6-7A, 6-7B, and 6-7C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright

compliance mechanism to a network of client and server computer systems.

FIGURE 6-8 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an embodiment of the present invention.

FIGURE 7-1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 7-2 is a block diagram of an exemplary network environment that can be utilized in accordance with an embodiment of the present invention.

FIGURE 7-3 is a block diagram of various exemplary functional components of a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 7-4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 7-5A is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with one embodiment of the present invention.

FIGURE 7-5B is a data flow block diagram showing an implementation of a component of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with another embodiment of the present invention.

FIGURE 7-5C is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files, in accordance with one embodiment of the present invention.

FIGURE 7-5D is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files through media file capture at a kernel level, in accordance with one embodiment of the present invention.

FIGURE 7-6A is a block diagram of an environment for preventing unauthorized copying of a media file, in accordance with one embodiment of the present invention.

FIGURES 7-7A, 7-7B, and 7-7C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright compliance mechanism to a network of client and server computer systems.

FIGURE 7-8 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an embodiment of the present invention.

FIGURE 8-1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 8-2 is a block diagram of an exemplary network environment that can be utilized in accordance with an embodiment of the present invention.

FIGURE 8-3 is a block diagram of various exemplary functional components of a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 8-4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 8-5A is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized recording of media files,

in accordance with one embodiment of the present invention.

FIGURE 8-5B is a data flow block diagram showing an implementation of a component of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with another embodiment of the present invention.

FIGURE 8-5C is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files, in accordance with one embodiment of the present invention.

FIGURE 8-5D is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files through media file capture at a kernel level, in accordance with one embodiment of the present invention.

FIGURE 8-6A is a block diagram of an environment for preventing unauthorized copying of a media file, in accordance with one embodiment of the present invention.

FIGURES 8-7A, 8-7B, and 8-7C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright compliance mechanism to a network of client and server computer systems.

FIGURE 8-8 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an

embodiment of the present invention.

FIGURE 8-9 is a block diagram of components of a copyright compliance mechanism installable from a media storage device upon which protected media files are disposed, in accordance with one embodiment of the present invention.

FIGURE 8-10 is a block diagram of a communicative environment for dynamic updating of a copyright compliance mechanism installed from a media storage device onto a client computer system, in accordance with one embodiment of the present invention.

FIGURE 8-11 is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized reproduction of a protected media file located on a media storage device, in accordance with one embodiment of the present invention.

FIGURE 9-1 is a block diagram of an exemplary computer system that can be utilized in accordance with an embodiment of the present invention.

FIGURE 9-2 is a block diagram of an exemplary network environment that

can be utilized in accordance with an embodiment of the present invention.

FIGURE 9-3 is a block diagram of various exemplary functional components of a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 9-4 is an illustration of an exemplary system for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention.

FIGURE 9-5A is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with one embodiment of the present invention.

FIGURE 9-5B is a data flow block diagram showing an implementation of a component of a copyright compliance mechanism for preventing unauthorized recording of media files, in accordance with another embodiment of the present invention.

FIGURE 9-5C is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files, in accordance with one embodiment of the present invention.

FIGURE 9-5D is a data flow block diagram showing an implementation of a copyright compliance mechanism for preventing unauthorized output of media files

through media file capture at a kernel level, in accordance with one embodiment of the present invention.

FIGURE 9-6A is a block diagram of an environment for preventing unauthorized copying of a media file, in accordance with one embodiment of the present invention.

FIGURES 9-7A, 9-7B, and 9-7C are a flowchart of steps performed in accordance with an embodiment of the present invention for providing a copyright compliance mechanism to a network of client and server computer systems.

FIGURE 9-8 is a diagram of an exemplary global media delivery system in which a copyright compliance mechanism can be implemented in accordance with an embodiment of the present invention.

FIGURE 9-9 is a block diagram of components of a copyright compliance mechanism installable from a media storage device upon which protected media files are disposed, in accordance with one embodiment of the present invention.

FIGURE 9-10 is a block diagram of a communicative environment for dynamic updating of a copyright compliance mechanism installed from a media storage device onto a client computer system, in accordance with one embodiment of the present invention.

FIGURE 9-11 is a data flow block diagram showing an implementation of a

copyright compliance mechanism for preventing unauthorized reproduction of a protected media file located on a media storage device, in accordance with one embodiment of the present invention.

FIGURE 9-12 is a block diagram of components of a usage compliance mechanism installable from a media storage device upon which protected media files are disposed, in accordance with one embodiment of the present invention.

FIGURE 9-13 is a block diagram of components of a usage compliance mechanism and content disposed on a media storage device, in accordance with one embodiment of the present invention.

FIGURE 9-14 is a block diagram of a communicative environment for controlling presentation of content on a media storage device, in accordance with one embodiment of the present invention.

FIGURE 9-15 is a data flow block diagram showing an implementation of a usage compliance mechanism for controlling presentation of content disposed on a media storage device, in accordance with one embodiment of the present invention.

FIGURE 9-16 is a flowchart of a process for controlling presentation of content disposed on a media storage device, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION:**Section 0: METHOD AND SYSTEM FOR CONTROLLING UNAUTHORIZED
REPRODUCTION OF PROTECTED MEDIA ON A MEDIA STORAGE
DEVICE**

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention

5 refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

10

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present

15 invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop, handheld, and/or other portable electronic device. For purposes of the present application, the term "portable electronic

20 device" is not intended to be limited solely to conventional handheld or portable computers. Instead, the term "portable electronic device" is also intended to include many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, satellite radios, and other personal digital devices.

Figure 1 is a block diagram illustrating an exemplary computer system 100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 100 can be nearly any type of computing system or electronic computing device

25

including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 100 and which are
5 executed by a processor(s) of computer system 100, in one embodiment. When executed, the instructions cause computer system 100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 100 of Figure 1 comprises an address/data bus 110 for communicating
10 information, one or more central processors 101 coupled to bus 110 for processing information and instructions. Central processor(s) 101 can be a microprocessor or any alternative type of processor. Computer system 100 also includes a computer usable volatile memory 102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 110 for
15 storing information and instructions for processor(s) 101. Computer system 100 further includes a computer usable non-volatile memory 103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc., coupled to bus 110 for storing static information and instructions for processor(s) 101. In one embodiment, non-volatile memory 103 can be
20 removable.

System 100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 104 coupled to bus 110 for enabling computer 100 to interface with other electronic devices. Communication interface 104 can include wired and/or wireless
25 communication functionality. For example, in one embodiment, communication interface 104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire

(IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, a satellite link, an Internet feed, a cable modem, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device

5 104. In such an instance, communication interface 104 may include a DSL modem.

Computer 100 of Figure 1 can also include one or more computer usable data storage device(s) 108 coupled to bus 110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 108 can be a magnetic storage

10 device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks or random array of inexpensive

15 discs) configuration. It is noted that data storage device 108 can be located internal and/or external of system 100 and communicatively coupled with system 100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 100. It is further noted that nearly any portable electronic device, e.g., device 100a, can also be communicatively coupled with system 100 via utilization of wired and/or wireless

20 technology, thereby expanding the functionality of system 100.

System 100 can also include an optional display device 105 coupled to bus 110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma

25 display, a field emission display (FED) or any other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 100 of Figure 1 further includes an optional alphanumeric input device 106 coupled to bus 110 for communicating information and command selections to processor(s) 101, in one embodiment. Alphanumeric input device 106 is coupled to bus 110 and includes alphanumeric and function keys. Also included in computer 100 is an optional cursor control device 107 coupled to bus 110 for communicating user input information and command selections to processor(s) 101. Cursor control device 107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, a optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 100 is configured with such functionality.

Figure 2 is a block diagram of an exemplary network 200 in which embodiments of the present invention may be implemented. In one example, network 200 enables one or more authorized client computer systems (e.g., 210, 220, and 230), each of which are coupled to Internet 201, to receive media content from a media content server, e.g., 251, via the Internet 201 while preventing unauthorized client computer systems from accessing media stored in a database of content server 251.

20

Network 200 includes a web server 250 and a content server 251 which are communicatively coupled to Internet 201. Further, web server 250 and content server 251 can be communicatively coupled without utilizing Internet 201, as shown. Web server 250, content server 251, and client computers 210, 220, and 230 can communicate with each other. It is noted that computers and servers of network 200 are well suited to be communicatively coupled in various implementations. For example, web server 250, content server 251, and client computer systems 210, 220, and 230 of network 200 can be communicatively coupled via wired

communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 2, it is noted that web server 250, content server 251, and client
5 computer systems 210, 220 and 230 of network 200 can, in one embodiment, be each
implemented in a manner similar to computer system 100 of Figure 1. However, the server and
computer systems in network 200 are not limited to such implementation. Additionally, web
server 250 and content server 251 can perform various functionalities within network 200. It is
also noted that, in one embodiment, web server 250 and content server 251 can both be disposed
10 on a single or a plurality of physical computer systems, e.g., computer system 100 of Figure 1.

Further, it is noted that network 200 can operate with and deliver any type of media
content, (e.g., audio, video, multimedia, graphics, information, data, software programs, etc.) in
any format. In one example, content server 251 can provide audio and video files to client
15 computers 210-230 via Internet 201.

Figure 3 is a block diagram of an exemplary copyright compliance mechanism (CCM)
300, for controlling distribution of, access to, and/or copyright compliance of media files, in
accordance with an embodiment of the present invention. In one embodiment, CCM 300
20 contains one or more software components and instructions for enabling compliance with
DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry
association of America) licensing agreements regarding media files. In one embodiment, CCM
300 may be integrated into existing and/or newly developed media player and recorder
applications. In another embodiment, CCM 300 may be implemented as stand alone but in
25 conjunction with existing media player/recorder applications, such that CCM 300 is
communicatively coupled to existing media player/recorder applications. Alternatively, CCM
300 can be installed as a stand alone mechanism within a client computer system 210.

Additionally, CCM 300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD (secure digital card), and/or as part of an installation package. In another example, CCM 300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD,
5 reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 300 may be installed on client system 210 in a clandestine manner, relative to a user.

There are currently two types of copyright licenses recognized by the DMCA for the protection of broadcast copyrighted material. One of the broadcast copyright licenses is a
10 compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one
15 example, the period of time can be three hours.

The other type of broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright holder grants permission for a server, e.g., web
20 server 250 and/or content server 251 of Figure 2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with "no save" license,
25 meaning that the end user is unable to save or store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user

to save that particularly music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This is commonly referred to as an interactive with "save" license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

5 It is noted that the DMCA allows for the "perfect" reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording.

10 In one embodiment, CCM (copyright compliance mechanism) 300 can be stored in web server 250 and/or content server 251 of network 200 and is configured to be installed into each client computer system, e.g., 210, 220 and 230, enabled to access the media files stored within content server 251 and/or web server 250. Alternatively, copyright compliance mechanism 300 can be externally disposed and communicatively coupled with a client computer system 200 via,
15 e.g., a portable media device 100a of Figure 1. In yet another embodiment, CCM 300 can be configured to be operable from a media storage device upon which media files may be disposed.

Copyright compliance mechanism 300 is configured to be operable while having portions of components, entire components, combinations of components, disposed within one or more
20 memory units and/or data storage devices of a computer system, e.g., 210, 220, and/or 230.

Additionally, portions of components, entire components and/or combinations of components of CCM 300 can be readily updated, e.g., via Internet 201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or licensing
25 agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications, or to counteract subversive and/or hacker-like attempts to unlawfully obtain one or more media files.

Referring to Figure 3, in one embodiment, CCM 300 is shown to include instructions 301 for enabling client computer system 210 to interact with web server 250 and content server 251 of network 200. Instructions 301 enable client computer system 210 to interact with servers, e.g., 250 and 251 in a network, e.g., 200.

The copyright compliance mechanism 300 also includes, in one embodiment, a user ID generator 302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 210. In one embodiment, the user ID and the cookie(s) are installed in computer system 210 prior to installation of the remaining components of the copyright compliance mechanism 300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 250 before the remaining components of a CCM 300 can be installed, within one embodiment of the present invention. Additionally, the user ID/user key can contain, but is not limited to, the user's name, the user's address, the user's credit card number, verified email address, and an identity (username) and password selected by the user.

Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 210, e.g., a MAC (machine address code) address, an IP address, and/or the serial number of the central processing unit (CPU) operable on computer system 210 and other information specific to the user and the computer system operated by the user. It is noted that the information regarding the client computer system, e.g., 210, the user of system 210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 210, web server 250 can determine when a user of one computer system, e.g., 210, has given their username and password to another user using another computer system, e.g., 220. Because the username, password, and the user's computer system 210 are closely associated, web server 250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 250. Web server 250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 3, copyright compliance mechanism 300 further includes one or more coder/decoders (codec) 303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 300. In the present embodiment, coder/decoder 303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 251.

In the present embodiment, coder/decoder 303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 300 rules are being enforced at any particular moment during media playback. It is noted that differing coder/decoders 303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 3, copyright compliance mechanism 300 also includes one or more agent programs 304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 210, 220, or 230, a server, e.g., web server 250 and/or content server 251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 304 can also be configured to maintain system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 300 also includes one or more system hooks 305, in one embodiment of the present invention. A system hook 305 is, in one embodiment, a library that is installed in a computer system, e.g., 210, and intercepts system wide events. For example, a system hook 305, in conjunction with skins 306, can govern certain properties and/or functionalities of media player applications operating within the client computer system, e. g., 210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during

playing of a media file, the progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that codec 303 compares the information for the media player application operating in client computer system, e.g., 210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 300 described herein.

The following C++ source code is exemplary implementation of the portion of a codec 303 for performing media player application detection, in accordance with an embodiment of the present invention. In another embodiment, the following source code can be modified to detect kernel streaming mechanisms operable within client system 210.

```

int
IsRecorderPresent(TCHAR * szAppClass,
20      TCHAR * szProdName)
{
    TCHAR      szWndText[_MAX_PATH]; /* buffer to receive title string for window */
    HWND      hWnd; /* handle to target window for operation */
    int      nRetVal; /* return value for operation */
25      /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
30      == 0)
    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
35      != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
}

```

```

    }
  }
  else
  {
5      /* attempt to locate window with specified class */
      if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
          != (HWND) 0)
      {
10          /* attempt to retrieve title string for window */
          if ( GetWindowText(hWnd,
                           szWndText,
                           _MAX_PATH)
              != 0)
          {
15              /* attempt to locate product name within title string */
              if ( _tcsstr(szWndText, szProdName)
                  != (TCHAR *) 0)
              {
20                  /* indicate application found */
                  nRetVal = 1;
              }
          }
      }
  }
25  /* return to caller */
  return nRetVal;
}

```

30 It is further noted that codec 303 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 210. For example, codec 303, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data pathways

35 that are commonly used for recording, such that the utilized data pathway is governed by the copyright compliance mechanism 300. In one embodiment, this can be performed within a driver shim, e.g., wave driver shim 309 of Figures 5A and 5B.

A driver shim can be utilized for nearly any software output device, e.g., a standard

40 Windows™ waveform output device, e.g., Windows™ Media Player, or hardware output device, e.g., speakers or headphones. Client computer system 210 is configured such that the driver

shim (e.g., 309 of Figures 5A, 5B, 5C, and 5D) will appear as the default waveform media device to client level application programs. Thus, requests for processing of waveform media input and/or output will pass through the driver shim prior to being forwarded to the actual waveform audio driver, media device driver 505 of Figures 5A and 5B. Such waveform

5 input/output suppression can be triggered by other components of CCM 300, e.g., agent 304, to be active when a recording operation is initiated by a client computer system, e.g., 210, during the play back of media files which are subject to the DMCA.

It is noted that alternative driver shims can be implemented for nearly any waveform

10 output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices, video, graphic and/or alphanumeric media files and video input and output devices.

15 The following C++ source code is an exemplary implementation of a portion of a codec 303 and/or a custom media device driver 307 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```

20  DWORD
    _stdcall
    widMessage(UINT      uDevId,
                UINT      uMsg,
                DWORD     dwUser,
25         DWORD     dwParam1,
                DWORD     dwParam2)
    {
        BOOL      bSkip;          /* flag indicating operation to be skipped */
        HWND      hWndMon;        /* handle to main window for monitor */
30         DWORD     dwRetVal;      /* return value for operation */

        /* initialize variables */
        bSkip = FALSE;
        dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;
35         if (uMsg == WIDM_START)

```

```

{
    /* attempt to locate window for monitor application */
    if ( (hWndMon = FindMonitorWindow())
        != (HWND) 0)
5      {
          /* obtain setting for driver */
          bDrvEnabled = ( SendMessage(hWndMon,
                                   uiRegMsg,
                                   0,
10                                   0)
                        == 0)
                        ? FALSE : TRUE;
      }

15      if (bDrvEnabled == TRUE)
      {
          /* indicate error in operation */
          dwRetVal = MMSYSERR_NOMEM;

20          /* indicate operation to be skipped */
          bSkip = TRUE;
      }
  }

25  if (bSkip == FALSE)
  {
      /* invoke entry point for original driver */
      dwRetVal = CallWinMessage(uDevId, uMsg, dwUser, dwParam1, dwParam2);
  }

30  /* return to caller */
  return dwRetVal;
}

```

35 It is noted that when properly configured, system hook 305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 210-230. In one embodiment, system hook 305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a Microsoft Windows™ operating environment, and accordingly can be

40 implemented in a variety of ways. However, it is also noted that system hook 305 can be readily adapted for implementation in alternative operating systems, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 3, copyright compliance mechanism 300 also includes one or more skins 306, which can be designed to be installed in a client computer system, e.g., 210-230. In one embodiment, skins 306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 306 are customizable
5 interfaces that, in one embodiment, are displayed on a display device (e.g., 105) of computer system 210 and provide functionalities for user interaction of delivered media content. Additionally, skins 306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

10

Furthermore, when system hook 305 is unable to govern a function of the media player application operable on a client computer system, e.g., 210, such that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 306 can be implemented to provide compliance.

15

Differing skins 306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 306a may be configured for utilization with a media content file protected under a non-interactive agreement (DMCA), such that skin 306a may not include a pause function, a stop function, a
20 selector function, and/or a save function, etc. Another skin, e.g., skin 306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with "no save" agreement (DMCA), such that skin 306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with "save" agreement, a save or a burn to CD function.

25

Still referring to Figure 3, it is further noted that in the present embodiment, each skin 306 can have a unique name and signature. In one embodiment, skin 306 can implemented, in

part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD5 hash, without knowledge of the MD5 hash table, changing the name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance mechanism 300 verifies skin 306, MD5 hash tables advantageously provide a deterrent against modifications made to the skin.

10 In one embodiment, copyright compliance mechanism 300 also includes one or more custom media device driver(s) 307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 210, can be configured to utilize a custom media device application, e.g., custom media device 310 (shown in Figure 5B, 5C, and 5D), to control unauthorized recording of media content files. A custom media device application can be, but is not limited to, a custom media audio device application for media files having sound content, a custom video device application for media files having graphical and/or alphanumeric content, etc. In one embodiment, custom media device 310 of Figure 5B is an emulation of the custom media device driver 307. With reference to audio media, the emulation is performed in a waveform audio driver associated with custom media device 310. Driver 307 is configured to receive a media file being outputted by system 210 prior to the media file being sent to a media output device, e.g., media output device 570, and/or a media output application, e.g., recording application 502. Examples of a media output device includes, but is not limited to, a video card for video files, a sound card for audio files, etc. Examples of a recording application can include, but is not limited to, CD burner applications for writing to another CDs, ripper applications which capture the media file and change the format of the media file, e.g., from a MP3 file to a .wav file. In one embodiment, client computer system 210 is configured with a custom media device driver 307 emulating

custom media device 310, and which is system 210's default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of skin 306 or a custom web based player application or as part of a CCM 300 installation bundle, for forcing or requiring system 210 to have driver 307 as the
5 default driver.

Therefore, when a media content file is received by system 210 from server 251, the media content file is playable, provided the media content file passes through the custom media device application (e.g., 310 of Figure 5B), emulated by custom media device driver 307, prior to
10 being outputted. However, if an alternative media player application is selected, delivered media files from server 251 will not play on system 210.

Thus, secured media player applications would issue a media request to the driver, e.g., 307, for the custom media device 310 which then performs necessary media input suppression,
15 e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 307 to a Windows™ waveform audio driver operable on system 210, thus
20 reducing instances of incompatibilities with existing media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 300, e.g., agents 304, system hooks 305, and skins 306, or a
25 combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of ways including, but limited to, those found at

developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 210 being configured with a custom media device as
5 the default device driver e.g., device 310 of Figures 5B, 5C, and 5D, emulated by a custom
media device driver 307, those media player applications that require their particular device
driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured
music. Further advantageous is that an emulated custom media device provides no native
support for those media player applications used as a recording mechanism, e.g., DirectSound
10 capture, (direct sound 504 of Figures 5A, 5B, 5C, and 5D) etc., that are able to bypass user-mode
drivers for most media devices. Additionally, by virtue of the media content being sent through
device driver 307, thus effectively disabling unauthorized saving/recording of media files, in one
embodiment, media files that are delivered in a secured delivery system do not have to be
encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-
15 encrypted media files utilizing less storage space and network resources than encrypted media
files, networks having limited resources can utilize the functionalities of driver 307 of CCM 300
to provide compliance with copyright restrictions and/or licensing agreements applicable with a
media content file without having the processing overhead of encrypted media files.

20 Figure 4 is an illustration of an exemplary system 400 for implementing a copyright
compliance mechanism in accordance with an embodiment of the present invention.
Specifically, system 400 illustrates web server 250, content server 251, or a combination of web
server 250 and content server 251 installing a copyright compliance mechanism (e.g., 300) in a
client's computer system (e.g., 210) for controlling media file distribution and controlling user
25 access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 210 can communicatively couple with a network (e.g., 200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 250 can initiate a registration process with the requesting client 210. Client registration can be accomplished in a variety of ways. For example, web server 250 may deliver to a client 210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 250 including, but not limited to, user's name, address, phone number, credit card number, verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 4, web server 250 can, in one embodiment, detect information related to the client's computer system, e.g., 210, and store that information in a user/media database 450. For example, web server 250 can detect a unique identifier of client computer system 210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network interface card) of client computer system 210 or the MAC address of the network interface adapter integrated on the motherboard of system 210. It is understood that a NIC enables a client computer system 210 to access web server 250 via Internet 201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 250 can, in one embodiment, detect and store (also in database 450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 450) within web server 250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 210, all or part of this information can be installed in client computer system 210. In one embodiment, client computer system 210 information can be in the form of a cookie. Web server 250 then verifies that the user and client computer system 210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 250 installs a copyright compliance mechanism (e.g., 300) into the client's computer system, e.g., 210, in one embodiment of the present invention. It is noted that web server 250 may not initiate installation of CCM 300 until the user ID, password, and client computer system 210 information is verified. A variety of common techniques can be employed to install an entire CCM 300, portions of components, entire components, and/or combinations or a function of components. For example, copyright compliance mechanism 300 can be installed in a hidden directory within client computer system 210, thereby preventing unauthorized access to it. In one embodiment of the present invention, it is noted that unless CCM 300 is installed in client computer system 210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 250 and/or content server 251,

Referring still to Figure 4, upon completion of client registration and installation of CCM 300, client computer system 210 can then request a media play list or a plurality of play lists, etc. In response, web server 250 determines whether the user of client computer system 210 is authorized to receive the media play list associated with the request. In one embodiment, web server 250 can request the username and password. Alternatively, web server 250 can utilize user database 450 to verify that computer 210 is authorized to receive a media play list. If client computer 210 is not authorized, web server 250 can initiate client registration, as described herein. Additionally, web server 250 can disconnect computer 210 or redirect it to an alternative

web site. Regardless, if the user and client computer system 210 are not authorized, web server 250 will not provide the requested play list to client computer system 210.

However, if client computer system 210 is authorized, web server 210 can check
5 copyright compliance mechanism 300 within data base 450 to determine if it, or any of the components therein, have been updated since the last time client computer system 210 logged in to web server 250. If a component of CCM 300 has been updated, web server 250 can install the updated component and/or a more current version of CCM 300 into client computer system 210, e.g., via Internet 201. If CCM 300 has not been updated, web server 250 can then deliver the
10 requested media play list to system 210 via Internet 201 along with an appended user key or user identification (ID). It is noted that user database 450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 210. Subsequently, the user of client computer system 210 can utilize the received media play list in combination with the media player application operating on system 210 to transmit a delivery
15 request for one or more desired pieces of media content from web server 250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 4, upon receiving the media content delivery request, web server 250 can then check the validity of the requesting media application and the attached user key. In
20 one embodiment, web server 250 can utilize user database 450 to check their validity. If either or both are invalid, web server 250, in one embodiment, can redirect unauthorized client computer system 210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 300 verifies that skins 306 are installed in client computer system 210. Additionally, CCM 300 further verifies that
25 system hook(s) 305 have been run or are running to govern certain functions of those media player applications operable within client computer system 210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 300 further diverts and/or

redirects certain pathways that are commonly used for recording, e.g., driver 307 of Figure 5A, device 310 of Figure 5B, device 570 of Figure 5C, and driver 505 of Figure 5D. Once CCM 300 has performed the above described functions, web server 250 then, in one embodiment, issues to the client computer 210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 210 receiving the redirect command from web server 250, the media player application operating on client computer system 210 automatically transmits a new request and the time sensitive access key to content server 251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 251. If invalid, unauthorized client computer 210 is redirected by content server 250 to protect against abuse of the system and unauthorized access to content server 251. If the time sensitive access key is valid, content server 251 retrieves the desired media content from content database 451 and delivers it to client computer system 210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g., title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 300 that identifies the media file as originating from a content

server 251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

5 Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or the entire CCM 300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to
10 modification of copyright compliance mechanism 300, modified portions of, or the entire updated CCM 300 can easily be installed in client computer system 210 in a variety of ways. For example, the updated CCM 300 can be installed during client interaction with web server 250, during user log-in, and/or while client computer system 210 is receiving the keyed play list.

15 Referring still to Figure 4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be stored in a database 450, content database 451 or in some
20 combination of databases 450 and 451. It is further noted that the messages being passed back and forth between client computer system 210 and web server 250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting with adds, public domain encryption
25 programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be

similarly encrypted. Furthermore, in another embodiment, the media files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 307.

- 5 Subsequent to media file decryption, the media file may be passed through CCM 300, e.g., a coder/decoder 303, to a media player application operating on client computer system 210, e.g. playback application 501 of Figures 5A, 5B, 5C, 5D, and 6A, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present
- 10 invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 306 of Figure 3. A skin 306 may be necessary when CCM 300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client computer system 210 to experience the media content.
- 15 Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 303 will
- 20 repeatedly ensure that CCM 300 rules are being enforced at any particular moment during media playback, shown as step 650 of Figure 6C.

- As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data
- !5 period, coder/decoder 303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the

presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These
5 presentation stoppages are collectively shown as step 651 of Figure 6C.

If the rules, in accordance with CCM 300, are enforced, the codec/decoder 303 retrieves a subsequent portion of the media content that is stored locally in client computer system 210. The newly retrieved portion of the media file is then presented by the client's media player
10 application. While the newly retrieved portion is presented, CCM 300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of
15 media files, CCM 300 can detect undesired activities and enforces those rules as defined by CCM 300.

Figure 5A is an exemplary logic/bit path block diagram 500A showing utilization of a wave shim driver, e.g., wave shim driver 309 of Figure 3, in conjunction with copyright
20 compliance mechanism 300, for selectively controlling recording of copyrighted media received by a client computer system, e.g., system 210, in one embodiment of the present invention. Copyright compliance mechanism 300 is, in one embodiment, installed and operational on client system 210 in the manner described herein.

25 In one embodiment, a copyright compliance mechanism 300 is shown as being communicatively coupled with a media playback application 501 via connection 520. Therefore, CCM 300 is enabled to communicate with playback application 501. In one embodiment, CCM

300 can be integrated into a media playback application. CCM 300 is also coupled to and controls a selectable switch 311 in wave shim driver 309 (as described in Figure 3) via connection 522. CCM 300 is further coupled to and controls a selectable switch 511 in direct sound 504 via connection 521. Depending upon the copyright restrictions and licensing
5 agreements applicable to an incoming media file, e.g., 499, CCM 300 controls whether switches 311 and 511 are open (shown), thus preventing incoming media 499 from reaching a media recording application, or closed (not shown) to allow recording of incoming media 499.

For example, incoming media 499 may originate from a content server, e.g., 251, coupled
10 to system 210. In another example, incoming media 499 may originate from a personal recording/electronic device, e.g., a MP3 player/recorder or similar device, coupled to system 210. Alternatively, incoming media 499 may originate from a magnetic, optical or alternative media storage device inserted into a media device player coupled to system 210, e.g., a CD or DVD inserted into a CD or DVD player, a hard disk in a hot swappable hard drive, an SD
15 (secure digital card) inserted into a SD reader, and the like. In yet another example, incoming media 499 may originate from another media player application or media recording application. Incoming media 499 may also originate from a satellite radio feed (e.g., XM radio), a personal communication device (e.g., a mobile phone), a cable television radio input (e.g., DMX (digital music express)), or a set-top box. It is noted that incoming media 499 can originate from nearly
20 any source that can be coupled to system 210. However, regardless of the source of incoming media 499, embodiments of the present invention, described herein, can prevent unauthorized recording of the media.

Figure 5A shows a media playback application 501, e.g., an audio, video, or other media
25 player application, operable within system 210 and configured to receive incoming media 499. Playback application 501 can be a playback application provided by an operating system, e.g., Media Player for Windows™ by Microsoft, a freely distributed playback application

downloadable from the Internet, e.g., RealPlayer or LiquidAudio, a playback application provided by a webcaster, e.g., PressPlay, or a playback application commercially available.

Figure 5A shows media device driver 505 which, in one implementation, may be a software driver for a sound card coupled to system 210 having a media output device 570, e.g., speakers or headphones, coupled therewith for media files having audio content. In another implementation, media device driver 505 may be a software driver for a video card coupled with a display device, e.g., 105, for displaying media files having alphanumeric and/or graphical content, and so on. With reference to audio files, it is well known that a majority of recording applications assume a computer system, e.g., 210, has a sound card disposed therein, providing full-duplex sound functionality to system 210. This means media output driver 505 can simultaneously cause playback and recording of incoming media files 499. For example, media device driver 505 can playback media 499 along wave-out line 539 to media output device 570 (e.g., speakers for audible playback) via wave-out line 580 while outputting media 499 on wave-out line 540 to eventually reach recording application 502.

For purposes of Figures 5A, 5B, 5C, and 5D, the terms wave-in line and wave-out line are referenced from the perspective of media device driver 505. Additionally, for the most part, wave-in lines are downwardly depicted and wave-out lines are upwardly depicted in Figures 5A, 5B, 5C, and 5D.

Continuing with Figure 5A, playback application 501 is coupled with an operating system (O/S) multimedia subsystem 503 and direct sound 504 via wave-in lines 531 and 551 respectively. O/S multimedia subsystem 503 is coupled to a wave shim driver 309 via wave-in line 533 and wave-out line 546. O/S multimedia subsystem 503 is also coupled to a recording application 502 via wave-out line 548. Operating system (O/S) multimedia subsystem 503 can be any O/S multimedia subsystem, e.g., a Windows™ multimedia subsystem for system 210

operating under a Microsoft O/S, a QuickTime™ multimedia subsystem for system 210 operating under an Apple O/S, and so on. Playback application 501 is also coupled with direct sound 504 via wave-in line 551.

5 Direct sound 504, in one instance, may represent access to a hardware acceleration feature in a standard audio device, enabling lower level access to components within media device driver 505. In another instance, direct sound 504 may represent a path that can be used by a recording application, e.g., Total Recorder, that can be further configured to bypass the default device driver, e.g., media device driver 505 to capture incoming media 499 for recording. For
10 example, direct sound 504 can be enabled to capture incoming media 499 via wave-in line 551 and unlawfully output media 499 to a recording application 502 via wave-out line 568, as well as media 499 eventually going to media device driver 505, the standard default driver.

 Still referring to Figure 5A, wave shim driver 309 is coupled with media device driver
15 505 via wave-in line 537 and wave-out line 542. Media device driver 505 is coupled with direct sound 504 via wave-in line 553 which is shown to converge with wave-in line 537 at media device driver 505. Media device driver 505 is also coupled with direct sound 504 via wave-out line 566.

20 Wave-out lines 542 and 566 are shown to diverge from wave-out line 540 at media device driver 505 into separate paths. Wave-out line 542 feeds into wave shim driver 309 and wave-out line 566 feeds into direct sound 504. When selectable switch 311 and 511 are open (shown), incoming media 499 cannot flow to recording application 502, thus preventing unauthorized recording of it.

25

 For example, incoming media 499 is received at playback application 501. Playback application 501 activates and communicates to CCM 300 regarding copyright restrictions and/or

licensing agreements applicable to incoming media 499. If recording restrictions apply to media 499, CCM 300 can, in one embodiment, open switches 311 and 511, thereby blocking access to recording application 502, effectively preventing unauthorized recording of media 499. In one embodiment, CCM 300 can detect if system 210 is configured with direct sound 504 selected as the default driver to capture incoming media 499, via wave-in line 551, or a recording application is detected and/or a hardware accelerator is active, such that wave driver shim 309 can be bypassed by direct sound 504. Upon detection, CCM 300 can control switch 511 such that the output path, wave-out line 568, to recording application 502 is blocked. It is further noted that CCM 300 can detect media recording applications and devices as described herein, with reference to Figure 3.

Alternatively, if media device driver 505 is selected as the default driver, incoming media 499 is output from playback application 501 to O/S multimedia subsystem 503 on wave-in line 531. From subsystem 503, media 499 is output to wave shim driver 309 via wave-in line 533. The wave shim driver 309 was described herein with reference to Figure 3. Media 499 is output from wave shim driver 309 to media device driver 505 via wave-in line 537. Once received by media device driver 505, media 499 can be output via wave-out line 539 to a media output device 570 coupled therewith via wave-out line 580. Additionally, media device driver 505 can simultaneously output media 499 on wave-out line 540 back to wave shim driver 309. Dependent upon recording restrictions applicable to media 499, CCM 300 can, in one embodiment, close switch 311 (not shown as closed), thereby allowing media 499 to be output from wave shim driver 309 to subsystem 503 (via wave-out line 546) and then to recording application 502 via wave-out line 548. Alternatively, CCM 300 can also open switch 311, thereby preventing media 499 from reaching recording application 502.

It is particularly noted that by virtue of CCM 300 controlling both switches 311 and 511, and therefore controlling wave-out line 548 and wave-out line 568 leading into recording

application 502, incoming media files, e.g., media 499, can be prevented from being recorded in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media. It is also noted that embodiments of the present invention in no way interfere with or inhibit the playback of incoming media 499.

5

Figure 5B is an exemplary logic/bit path block diagram 500B of a client computer system, e.g., 210, configured with a copyright compliance mechanism 300 for preventing unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 300 is, in one embodiment, coupled with and
10 operational on client system 210 in the manner with reference to Figures 4, 5A, 5C, 5D, 6, and 7.

Diagram 500B of Figure 5B is similar to diagram 500A of Figure 5A, with a few changes. Particularly, diagram 500B includes a custom media device 310 communicatively interposed between and coupled to O/S multimedia subsystem 503 and wave shim driver 309.
15 Custom media device 310 is coupled to O/S multimedia subsystem via wave-in line 533 and wave-out line 546. Custom media device 310 is coupled with wave shim driver 309 via wave-in line 535 and wave-out line 544. Additionally, custom media device 310 is coupled with direct sound 504 via wave-in line 553 which converges with wave-in line 533 and wave-out line 566 which diverges from wave-out line 546, in one embodiment.

20

Also added to Figure 5B is a media hardware output device 570 that is coupled to media device hardware driver 505 via line 580. Media hardware output device 570 can be, but is not limited to, a sound card for audio playback, a video card for video, graphical, alphanumeric, etc, output, and the like.

25

In one embodiment, CCM 300 is communicatively coupled with playback application 501 via connection 520, waveform driver shim 309 via connection 522, and custom media device

310, via connection 521. CCM 300 is coupled to and controls a selectable switch 311 in waveform driver shim 309 via connection 522. CCM 300 is also coupled to and controls a selectable switch 312 in custom audio device 310 via connection 521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 5 499, CCM 300 controls whether switches 311 and 312 are open (shown), thus preventing the incoming media 499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 499.

Continuing with Figure 5B, direct sound 504 is shown coupled with custom media device 10 310 via wave-in line 553, instead of being coupled with media device driver 505 (Figure 5A). In one embodiment, custom audio device 310 mandates explicit selection through system 210, meaning that custom audio device 310 needs to be selected as a default driver of system 210. By virtue of having the selection of custom media device 310 as the default driver of system 210, the data path necessary for direct sound 504 to capture the media content is selectively closed.

15

For example, incoming media 499 originating from nearly any source with reference to Figure 5A is received by media playback application 501 of system 210. Playback application 501 communicates to CCM 300, via connection 520, to determine whether incoming media 499 is protected by any copyright restrictions and/or licensing agreements. Playback application 501 20 communicates with CCM 300 to control switch 311 and 312 accordingly. In the present example, recording of incoming media 499 would violate applicable restrictions and/or agreements and therefore switch 312 is in an open position, such that the output path to recording application 502, e.g., wave-out line 548 and/or wave-out line 568, is effectively blocked, thereby preventing unauthorized recording of media 499.

25

Alternatively, if media device driver 505 is selected as the default driver, incoming media 499 continues from O/S multimedia subsystem 503, through custom audio device 310, wave

driver shim 309, and into media device driver 505 where media 499 can be simultaneously output to media output device 570 via line 580, and output on wave-out line 540 to wave-and outputted by media device driver 505 to wave shim driver 309 on wave-out line 542. However, by virtue of CCM 300 controlling switch 311, wave-out line 544 which eventually leads to
5 recording application 502 is blocked, thus effectively preventing unauthorized recording of media 499.

It is particularly noted that by virtue of CCM 300 controlling both switches 311 and 312 and therefore controlling wave-out line 548 and wave-out line 568, any incoming media files,
10 e.g., incoming media 499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 5B, it is further noted that custom media device 310 allows for
15 unfettered playback of incoming media 499. Additionally, at any time during playback of media 499, custom media device 310 can be dynamically activated by CCM 300.

Figure 5C is an exemplary logic/bit path block diagram 500C of a client computer system, e.g., 210, configured with a copyright compliance mechanism 300 for preventing
20 unauthorized output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 300 is, in one embodiment, coupled with and operational on client system 210 in the manner with reference to Figures 4, 5A, 5B, 5D, 6, and 7.

25 Diagram 500C of Figure 5C is similar to diagram 500B of Figure 5B, with a few changes. Particularly, diagram 500C includes a media hardware output device 570 that is coupled with a media device driver 505. In one embodiment, media hardware output device 570 can be a

S/PDIF (Sony/Phillips Digital Interface) card for providing multiple outputs, e.g., an analog output 573 and a digital output 575. An alternative media hardware output device providing similar digital output can also be implemented as device 570 including, but not limited to, a USB (universal serial bus) output device and/or an externally accessible USB port located on system 210, a FireWire (IEEE1394) output device and/or an externally accessible FireWire port located on system 210, with wireline or wireless functionality. In the present embodiment, media hardware output device 570 is shown to include a switch 571 controlled by CCM 300 via communication line 523, similar to switches 311 and 312, for controlling output of incoming media 499.

In one embodiment, CCM 300 is communicatively coupled with playback application 501 via connection 520, waveform driver shim 309 via connection 522, custom media device 310, via connection 521, and media hardware output device 570 via connection 523. CCM 300 is coupled to and controls a selectable switch 311 in waveform driver shim 309 via connection 522. CCM 300 is also coupled to and controls a selectable switch 312 in custom audio device 310 via connection 521. CCM 300 is further coupled to and controls a selectable switch 571 in media hardware output device 570 via connection 523. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 499, CCM 300 controls whether switches 311 and 312 are open (shown), thus preventing the incoming media 499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 499. Additionally, CCM 300 controls whether switch 571 is open (shown), thus preventing incoming media 499 from being output from digital output 575 of media hardware output device 570, or closed (not shown) to allow incoming media 499 to be output from media hardware output device 570.

By controlling media hardware output device 570, copyright compliance mechanism 300 can prevent unauthorized output of incoming media 499 to, e.g., a digital recording device that

may be coupled with digital output 575 of media hardware output device 570. Accordingly, in one embodiment, CCM 300 is enabled to also detect digital recording devices that may be coupled to a digital output line, e.g., 571, of a media hardware output device, e.g., 570.

Examples of a digital recording device that can be coupled to media hardware output device 570 can include, but is not limited to, mini-disc recorders, MP3 recorders, personal digital recorders, 5 digital recording devices coupled with multimedia systems, personal communication devices, set-top boxes, and/or nearly any digital device that can capture an incoming media 499 being output from a media hardware output device 570, e.g. a sound card.

10 Continuing with Figure 5C, direct sound 504 is shown coupled with custom media device 310 via wave-in line 553, instead of being coupled with media device driver 505 (Figure 5A). In one embodiment, custom audio device 310 mandates explicit selection through system 210, meaning that custom audio device 310 is needs to be selected as a default driver of system 210. By virtue of having the selection of custom media device 310 as the default driver of system 210, 15 the data path necessary for direct sound 504 to capture the media content is selectively closed.

For example, incoming media 499 originating from nearly any source with reference to Figure 5A is received by media playback application 501 of system 210. Playback application 501 communicates to CCM 300, via connection 520, to determine whether incoming media 499 20 is protected by any copyright restrictions and/or licensing agreements. Playback application 501 communicates with CCM 300 to control switch 311, 312, and 571 accordingly. In the present example, recording of incoming media 499 would violate applicable restrictions and/or agreements and therefore switch 312 is in an open position, such that the output path to recording application 502, e.g., wave-out line 548 and/or wave-out line 568, is effectively blocked, thereby 25 preventing unauthorized recording of media 499.

Alternatively, if media device driver 505 is selected as the default driver, incoming media 499 continues from O/S multimedia subsystem 503, through custom audio device 310, wave driver shim 309, and into media device driver 505 where media 499 can be simultaneously output to media output device 570 via line 580, and output on wave-out line 540 to wave-and
5 outputted by media device driver 505 to wave shim driver 309 on wave-out line 542. However, by virtue of CCM 300 controlling switch 311, wave-out line 544 which eventually leads to recording application 502 is blocked, thus effectively preventing unauthorized recording of media 499.

10 It is particularly noted that by virtue of CCM 300 controlling both switches 311 and 312 and therefore controlling wave-out line 548 and wave-out line 568, any incoming media files, e.g., incoming media 499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

15 Still referring to Figure 5C, it is particularly noted that although CCM 300 can prevent unauthorized recording of incoming media 499 by controlling switches 311 and 312, thus preventing incoming media 499 from reaching recording application 502, controlling switches 311 and 312 do nothing to prevent incoming media 499 from being captured by a peripheral
20 digital device, e.g., a mini-disc recorder, etc., coupled to a digital output 575 of device 570. Thus, by also controlling the output, via digital output 575 of media hardware output device 570, through control of switch 571, CCM 300 can prevent unauthorized capturing of incoming media 499 during output, e.g., on a sound card for audio files, a video card for video and/or graphical files, regardless of whether incoming media 499 is received in a secure and encrypted manner.
25 However, when switch 571 is in a closed position, incoming media 499 may be played back in an unfettered manner. Additionally, at any time during playback of media 499, switch 312 of

custom media device 310, switch 311 of media device driver 309, and/or switch 571 of media hardware output device 570 can be dynamically activated by CCM 300.

Figure 5D is an exemplary logic/bit path block diagram 500D of a client computer system, e.g., 210, configured with a copyright compliance mechanism 300 for preventing unauthorized kernel based output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 300 is, in one embodiment, coupled with and operational on client system 210 in the manner described herein with reference to Figures 4, 5A, 5B, 5C, 6, and 7.

Diagram 500D of Figure 5D is similar to diagram 500C of Figure 5C, with some changes. Particularly, diagram 500D includes a kernel streaming mechanism 515, e.g., DirectKS, that is coupled with a media device driver 505. In one embodiment, DirectKS 515 can be used for establishing a direct connection with media device driver 505. In the present embodiment, media device driver 505 is shown to include a switch 511 controlled by CCM 300 via communication line 524, that is similar to switches 311, 312, and 571, for controlling output of incoming media 499.

In one embodiment, CCM 300 is communicatively coupled with: playback application 501 via connection 520, waveform driver shim 309 via connection 522, custom media device 310, via connection 521, and media device driver 505 via connection 524. Specifically, CCM 300 is coupled to and controls a selectable switch 311 in waveform driver shim 309 via connection 522. CCM 300 is also coupled to and controls a selectable switch 312 in custom audio device 310 via connection 521. CCM 300 is further coupled to and controls a selectable switch 511 in media device driver 505 via connection 524. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 499, CCM 300 controls whether switches 311 and 312 are open (shown), thus preventing the

incoming media 499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 499. Additionally, CCM 300 controls whether switch 511 is open (shown), thus preventing incoming media 499 from being returned from media device driver 505 to playback application 501, where DirectKS 515 can capture incoming media 499 and redirect it to recording application 502 to create an unauthorized copy or recording of incoming media 499. CCM 300 can also control whether switch 511 is closed (not shown) to allow incoming media 499 to be returned to playback application 501, where DirectKS 515 can capture and redirect incoming media 499 to recording application 502.

DirectKS 515, in one embodiment, may represent a kernel streaming mechanism that is adapted to establish a direct connection with a media device driver 505 of an operating system operable on client computer system 210, enabling kernel level access to media device driver 505. A kernel streaming mechanism can be implemented for the purpose of precluding utilization of standard audio APIs (application programming interfaces) to play or record media content, with particular attention paid to those playback applications with low latency requirements. DirectKS 515 can bypass existing APIs and communicate with media device driver 505. DirectKS 515 can be readily adapted to work in conjunction with a playback application, e.g., 501, to capture and redirect incoming media 499 to recording application 502, via wave-out line 588. Accordingly, DirectKS 515 can be implemented to create unauthorized media recordings.

By controlling media device driver 505, copyright compliance mechanism 300 can prevent unauthorized output of incoming media 499 to, e.g., a digital recording device 529 that may be coupled with recording application 502. In one embodiment, media device driver 505 is configured through the kernel mixer (not shown) to control the data path. Additionally, in one embodiment, CCM 300 is enabled to also detect a kernel streaming mechanism 515 (e.g., DirectKS) that may be operable on client computer system 210, as described herein with reference to Figure 3.

In one embodiment, custom media device 310 mandates explicit selection through system 210, meaning that custom media device 310 is needs to be selected as a default driver of system 210. By virtue of having the selection of custom media device 310 as the default driver of
5 system 210, the data path necessary for direct sound 504 to capture the media content is selectively closed.

For example, incoming media 499 originating from nearly any source with reference to Figure 5A is received by media playback application 501 of system 210. Playback application
10 501 communicates to CCM 300, via connection 520, to determine whether incoming media 499 is protected by any copyright restrictions and/or licensing agreements. Playback application 501 communicates with CCM 300 to control switches 311, 312, 571, and 511, accordingly. In the present example, recording of incoming media 499 would violate applicable restrictions and/or
15 agreements and therefore switch 511 is in an open position, such that the output path to recording application 502, e.g., wave-out line 548 and/or wave-out line 568 and/or wave-out line 588, is effectively blocked, thereby preventing unauthorized recording of media 499.

Still referring to Figure 5D, it is particularly noted that although CCM 300 can prevent unauthorized recording of incoming media 499 by controlling switches 311, 312, and 571, thus
20 preventing incoming media 499 from reaching recording application 502, controlling switches 311, 312, and 571, do nothing to prevent incoming media 499 from being returned to recording application 502 by a kernel streaming mechanism 515(e.g., DirectKS), which enables capturing and redirecting of incoming media 499 to recording application 502, via wave-out line 588. Thus, by also controlling switch 511 of media device driver 505, CCM 300 can prevent kernel
25 streaming mechanism 515 from returning incoming media 499 to recording application 502, thereby preventing incoming media 499 from being captured and redirected to recording application 502 in an attempt to create and unauthorized copy and/or recording of incoming

media 499. However, when switch 511 is in a closed position, incoming media 499 may be returned to a recording application 502, such that recording could be possible, provided recording does not violate copyright restrictions applicable to incoming media 499.

Additionally, at any time during playback of media 499, switch 312 of custom media device 310, switch 311 of wave shim driver 309, and/or switch 511 of media device driver 505 can be dynamically activated by CCM 300.

Figure 6A is an block diagram of a media file, e.g., incoming media 499, adapted to be received by a playback application, e.g., 501 of Figures 5A, 5B, 5C, and 5D, configured with an indicator 605 for enabling incoming media 499 to comply with rules according to the SCMS (serial copy management system). When applicable to a media file, e.g., 499, the SCMS allows for one copy of a copyrighted media file to be made, but not for copies of copies to be made. Thus, if incoming media 499 can be captured by a recording application, e.g., 502 of Figures 5A, 5B, 5C, and/or 5D, and/or a recording device, e.g. 529, and/or a peripheral recording device and/or a recording application coupled to a digital output of a media hardware output device, e.g., digital output 575 of media hardware output device 570 of Figures 5B, 5C, and 5D, and/or a kernel streaming mechanism 515, e.g., DirectKS of Figure 5D, unauthorized copying and/or recording may be accomplished.

Playback application 501 is coupled with CCM 300 via communication line 520 in a manner analogous to Figures 5A, 5B, 5C, and/or 5D. Although not shown in Figure 6, it is noted that CCM 300 is also coupled to switches 311 and 511 as shown in Figure 5A, switches 311 and 312 in Figure 5B, switches 311, 312, and 571 in Figure 5C, and switches 312, 311, 571, and 511, in Figure 5D.

In one embodiment, an indicator 605 is attached to incoming media 499 for preventing unauthorized copying or recording in accordance with the SCMS. In one embodiment, indicator

605 can be a bit that may be transmitted prior to beginning the delivery of incoming media 499 to playback application 501. In another embodiment, indicator 605 may be placed at the beginning of the bit stream of incoming media 499. In another embodiment, indicator 605 may be placed within a frame period of incoming media 499, e.g., every fifth frame, or any other desired frame
5 period. In another embodiment, indicator 605 may be transmitted at a particular time interval or intervals during delivery of the media file, e.g. incoming media 499. Thus, indicator 605 may be placed nearly anywhere within or attached to the bit stream related to incoming media 499.

Indicator 605 may be comprised of various indicators, e.g., a level 0 indicator, a level 1
10 indicator, and a level 2 indicator, in one embodiment of the present invention. In the present embodiment, a level 0 indicator may be for indicating to CCM 300 that copying is permitted without restriction, e.g., incoming media 499 is not copyrighted or that the copyright is not asserted. In the present embodiment, a level 1 indicator may be for indicating to CCM 300 that one generation of copies of incoming media 499 may be made, such that incoming media 499 is
15 an original copy and that one copy may be made. In the present embodiment, a level 2 indicator may be for indicating to CCM 300 that incoming media 499 is copyright protected and/or a copy thereof, and as such no digital copying is permitted.

For example, incoming media 499 is received by playback application 501. Application
20 501 detects an indicator 605 attached therewith, in this example, a level 2 bit is placed in the bit stream for indicating to CCM 300 that copying is not permitted.

For example, when CCM 300 is configured in system 210 such as that shown in Figure 5A, in response to a level 2 indicator bit, CCM 300, while controlling the audio path, then
25 activates switches 311 and 511 to prevent any recording of incoming media 499.

When CCM 300 is configured in system 210 such as that shown in Figure 5B, in response to a level 2 indicator bit, CCM 300, while controlling the audio path, then activates switches 311 and 312 to prevent any recording of incoming media 499.

- 5 When CCM 300 is configured in system 210 such as that shown in Figure 5C, in response to a level 2 indicator bit, CCM 300, while controlling the audio path, then activates switches 311, 312, and 571 to prevent any recording of incoming media 499.

- 10 It is noted that CCM 300 can activate or deactivate switches coupled therewith, as described herein with reference to Figures 5A, 5B, 5C, and 5D, thereby funneling incoming media 499 through the secure media path, in this instance the audio path, to prevent unauthorized copying of incoming media 499. It is further noted that CCM 300 can detect media recording applications and devices as described herein, with reference to Figure 3.

- 15 Figures 7A, 7B, and 7C, are a flowchart 700 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 700 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable
20 instructions reside, for example, in data storage features such as computer usable volatile memory 104 and/or computer usable non-volatile memory 103 of Figure 1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 700, such steps are exemplary. That is, the present invention is well suited to performing various other steps or
25 variations of the steps recited in Figures 7A, 7B, and 7C. Within the present embodiment, it should be appreciated that the steps of flowchart 700 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for restricting recording of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 300, can be installed in the client computer system 210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 700 is described in conjunction with Figures 2, 3, 4, 5A, 5B, 5C, and 5D, in order to more fully describe the operation of the present embodiment. In step 702 of Figure 7A, a user of a computer system, e.g., 210, causes the computer to communicatively couple to a web server, e.g., 250, via one or more communication networks, e.g., Internet 201, and proceeds to attempt to log in. It is understood that the log in process of step 702 can be accomplished in a variety of ways in accordance with the present invention.

In step 704 of Figure 7A, web server 250 accesses a user database, e.g., 450, to determine whether the user and the computer system 210 logging in are registered with it. If the user and

computer system 210 are registered with web server 250, the present embodiment proceeds to step 714. However, if the user and computer system 210 are logging in for the first time, web server 250 can initiate a user and computer system 210 registration process at step 706.

5 In step 706, registration of the user and computer system 210 is initiated. The user and computer system registration process can involve the user of computer system 210 providing personal information including, but not limited to, their name, address, phone number, credit card number, and the like. Web server 250 can verify the accuracy of the information provided. Web server 250 can also acquire information regarding the user's computer system 210
10 including, but not limited to, identification of media players disposed and operable on system 210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address. Additionally, web server 250 can further request that the user of computer system 210 to select a username and password.

15 In step 708 of Figure 7A, subsequent to the completion of the registration process, web server 250 generates a unique user identification (ID) or user key associated with the user of client computer system 210. The unique user ID, or user key, is then stored by web server 250 in a manner that is associated with that registered user. Furthermore, one or more cookies
20 containing that information specific to that user and the user's computer system 210, is installed in a non-volatile memory device, e.g., 103 and/or data storage device 108 of computer system 210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user
25 key, has been previously generated for the user and computer 210 that initially logged-in at step 702, the present embodiment proceeds to step 714

In step 710, web server 250 verifies that the user ID and the cookie(s) are properly installed in computer system 210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 250 can re-initiate the registration process at
5 step 706. Alternatively, web server 250 can decouple computer system 210 from the network, thereby requiring a re-log in by the user of computer 210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 712.

In step 712 of Figure 7A, web server 250 can install a version of a copyright compliance
10 mechanism, e.g., 300, onto one or more non-volatile memory devices of computer system 210. Installing CCM 300 into user's computer system 210 can facilitate client side compliance with licensing agreements and copyright restrictions applicable to specific delivered copyrighted media content. At step 712, the components of CCM 300, such as instructions 301,
15 coder/decoder (codec) 303, agent programs 304, system hooks 305, skins 306, and custom media device drivers 307 (e.g., custom media device 310 of Figures 5B, 5C, and 5D), are installed in computer system 210, such as that shown in Figures 5A, 5B, 5C, and 5D. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 300 into computer system 210. However, step 712 is well suited to install CCM 300 on computer system 210 in a wide variety of ways in accordance with the present embodiment. For example, CCM 300 can
20 be installed as an integrated component within a media player application, media recorder application, and/or media player/recorder applications. Alternatively, CCM 300 can be installed as a stand alone mechanism within a client computer system 210. Additionally, CCM 300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD, and/or as part of an installation package. In another
25 embodiment, CCM 300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD, reading a document, viewing a video, etc.

It is noted that, in one embodiment, CCM 300 may be installed on client system 210 in a clandestine manner, relative to a user.

In step 714, web server 250 can request the previously established username and password of the user of client computer system 210. Accordingly, the user of client computer system 210 causes it to transmit to web server 250 the previously established username and password. Upon the receipt thereof, web server 250 may access a user database, e.g., 450, to determine their validity. If the username and password are invalid, web server 250 refuses access wherein flowchart 500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 716.

In step 716 of Figure 7A, web server 250 can access media file database 450 to determine if copyright compliance mechanism 300 has been updated to reflect changes made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 300 can provide compliance with current copyright restrictions.

Continuing with step 716, if web server 250 determines that CCM 300, or components thereof, of computer 210 has been updated, web server 250 initiates installation of the newer components and/or the most current version of CCM 300 into computer system 210, shown as step 718. If web server 250 determines that the current version of CCM 300 installed on system 210 does not have to be updated, the present embodiment proceeds to step 720 of Figure 7B.

In step 720 of Figure 7B, the user of client computer system 210 causes it to transmit to web server 250, e.g., via Internet 201, a request for a play list of available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g.,
5 251.

In step 722, in response to web server 250 receiving the play list request, web server 250 transmits to client computer system 210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media
10 content play list in a manner invisible to the user. It is noted that the media content in content server 251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, and the like. The media content play list of step 720 can be implemented in diverse ways. In one example, web server 250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media
15 content titles, or different lists of titles, can be loaded from content server 251 and passed to a CGI (common gateway interface) program operating on web server 250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 210. It is understood that the CGI can be written in nearly any software computing language.

20

In step 724 of Figure 7B, the user of client computer system 210 can utilize the received media content play list in conjunction with a media player application in order to cause client computer system 210 to transmit a request to web server 250 for delivery of desired media content, and wherein the user ID is automatically included therewith. The media content play
25 list provided to client computer system 210 by web server 250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to

client computer system 250 and the order in which the content will be delivered. Additionally, the user of client computer system 250 can create one or more customized play lists and store those play lists in system 250 and/or within web server 250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions thereof. In another embodiment, the received media content play list can include a random media content delivery choice that the user of client computer system 210 can transmit to web server 250, with the user ID, to request delivery of the media content in a random manner.

In step 726, upon receiving the request for media content from client computer system 210, web server 250 determines whether the requesting media application operating on client computer system 210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 250 determines that the media application operating on system 210 is not a valid media application, the present embodiment proceeds to step 727 which in one embodiment, redirects client computer system 210 to a web site where the user of system 210 can download a valid media player application or to a software application which can identify client computer system 210, log system 210 out of web server 250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 250 determines that the media application operating on system 210 is a valid media application, the present embodiment proceeds to step 728.

25

In step 728 of Figure 7B, the present embodiment causes web server 250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client

computer system 210 is valid. If web server 250 determines that the user ID is invalid, the present embodiment proceeds to step 729 where client computer system 210 can be logged off web server 250 or client computer system 250 can be returned to step 706 (of Figure 7A) to re-register and to have another unique user ID generated by web server 250. It is noted that the order in which steps 726 and 728 are performed can be altered such that step 728 can be performed prior to step 726. If web server 250 determines that the user ID is valid, the present embodiment proceeds to step 730.

In step 730, prior to web server 250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 732, CCM 300 governs certain media player applications and/or functions thereof that are operable on client computer system 210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 300 can utilize system hooks 305 to accomplish the functionality of step 730.

In step 732 of Figure 7C, the present embodiment causes web server 250 to transmit to client computer system 210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 251. The address is time sensitive because, in one embodiment, the content server 251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 251 that are actively coupled to one or more client computer systems (e.g., 210 - 230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client computer systems are uncoupled from the media content source of content server 251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 210 to content server 251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 251), or by routing a requesting client computer system (e.g., 210, 220, or 230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 210 - 230.

15

The functionality of step 732 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

20

In step 734 of Figure 7C, upon receiving the redirection command, the present embodiment causes the media playback application 501 (Figures 5A, 5B, 5C, and 5D) operating on client computer system 210 to automatically transmit to content server 251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

25

In step 726 of Figure 7C, content server 251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 251 determines that the time sensitive access key is valid, the present embodiment proceeds to step 738 of Figure 7C. However, if content server 251 determines that the time access key is not valid, the present
5 embodiment proceeds to step 737, a client redirect.

In step 737, content server redirects client computer 210 to step 732 (not shown) where a new access key is generated. Alternatively, step 737 causes the present embodiment to return to step 704 of Figure 7A. In yet another embodiment, step 737 causes client computer system 210
10 to be disconnected from content server 251.

In step 738 of Figure 7C, content server 251 transmits the requested high fidelity media content to client computer system 210. It is noted that each media content file delivered to client computer system 210 can have a header attached thereto, prior to delivery, as described with
15 reference to Figure 4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can
20 be utilized.

Still referring to step 738, content server 251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 210 as fast as the network transfer rate allows. Further, content server 251 can have its
25 download rate adapted to be equal to the transfer rate of the network to which it is coupled. In another embodiment, the content server 251 download rate can be adapted to equal the network transfer rate of the client computer system 210 to which the media content is being delivered.

For example, if client computer system 210 is coupled to Internet 201 via a T1 connection, then content server 251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 210, content server 251 is then able to transmit requested media content to another client
5 computer system, e.g., 220 or 230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 250 to client
10 computer system 210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 300 to client 210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 251 can transmit to client computer system 250
15 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 740 of Figure 7C, upon receiving the requested high fidelity media content from content server 251, the present embodiment causes client computer system 210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content
20 is stored in client computer system 210 in a manner that restricts unauthorized redistribution. For example, the present embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time. Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory
25 device, e.g., 103 or data storage device 108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is

further noted that in one embodiment, when a user of client computer system 210 turns the computer off or causes client computer system 210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

5 Still referring to step 740, in another embodiment, the present embodiment can cause client computer system 210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media
10 content can be stored in a manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is noted that by storing media content with client computer system 210 (when allowed by applicable licensing agreements and copyright restrictions), content server 251 does not need to redeliver the same
15 media content to client computer system 210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

 In step 742 of Figure 7C, the received media content file is then fed into a media player application (e.g., playback application 501 of Figures 5A, 5B, 5C, and 5D), which then runs it
20 through a codec, e.g., coder/decoder 303 of CCM 300, in one embodiment. In response, coder/decoder 303 sends an authorization request to the server, e.g., 251, with attached authorization data, as described herein. In response to receiving codec's 303 authorization request, server 251 compares the received authorization data with that stored in server 251, and subsequently, the present embodiment proceeds to step 744.

25

 In step 744, the server 251 responds with a pass or fail authorization. If server 251 responds with a fail, such that the received authorization data is invalid, the present method can

proceed to step 745, where server 251 can, in one embodiment, notify the user of client system 210, e.g., by utilization of skin 306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented to the user of client computer system 210, thereby informing the user that the delivery
5 failed. However, if the authorization data passes, the present method proceeds to step 746.

In step 746, server 251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file via media playback application 501 of Figures 5A, 5B, 5C, and 5D. In one embodiment, a decryption key can be
10 included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 748.

In step 748 of Figure 6C, subsequent to media file decryption, the media file may be
15 passed through CCM 300, e.g., a coder/decoder 303, to a media player application operating on client computer system 210, e.g., playback application 501 of Figures 5A, 5B, 5C, and 5D, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be involved in order to
20 experience the media content, e.g., skin 306 of Figure 3. Skin 306 may be implemented when CCM 300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard media player can be utilized by client computer system 210 to experience
25 the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating

system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, coder/decoder 303 will repeatedly ensure that CCM 300 rules are being enforced at any particular moment during media playback, shown as step 750.

5

In step 750, as the media file content is delivered to the media player application, e.g., media player application 501 of Figures 5A, 5B, 5C, and 5D, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 300. If the rules are not enforced, e.g., change due to a user opening up a recording application (e.g., Total Recorder or alternative application) the present method proceeds to step 751. If the rules, in accordance with CCM 300, are enforced, the present method then proceeds to step 752.

15 In step 751 of Figure 7C, if the rules according to CCM 300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In one embodiment, CCM 300 can selectively control switches 311 and 511 (Figure 5A) to prevent output of incoming media 499 (Figures 5A, 5B, 5C, and 5D) to a recording application 502 (Figures 5A, 5B, and 5C, via wave shim driver 309 and direct sound 504 respectively, thus preventing unauthorized recording of incoming media 499. In another embodiment, CCM 300 can selectively control switches 311 and 312 (Figure 5B) to prevent output of incoming media 499 to recording application 502 via wave shim driver 309 and custom media device 310, thus preventing unauthorized recording of incoming media 499. In yet another embodiment, CCM 300 can selectively control switches 311, 312, to not only prevent incoming media 499 from being recorded in an unauthorized manner but can also selectively control switch 571 (Figure 5C) to prevent unauthorized output of incoming media 499 via digital output 575 of media hardware output device 570. In yet another embodiment, CCM 300 can selectively control

switches 311, 312, 571, and 511 to a prevent kernel streaming mechanism 515, e.g., DirectKS of Figure 5D, which can establish a connection with media device driver 505 of Figure 5D, from capturing incoming media content and returning it to a recording application (e.g., 502) to create an unauthorized recording of the media content. In one embodiment, incoming media 499 may not be output from digital output 575. In another embodiment, incoming media 499 may be output via digital output 575 but in an inaudible manner, e.g., silence. In yet another embodiment, incoming media 499 be audible but recording functionality can be disabled, such that the media content cannot be recorded.

10 In step 752, if the rules are enforced in accordance with CCM 300, coder/decoder 303 retrieves a subsequent portion of the media content that is stored locally in client computer system 210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the present method as step 748. While the newly retrieved portion is presented, embodiments of the present method then again perform step 750, then step 752 or 15 751, then step 748, then 750, etc., in a continual loop until the media file contents are presented in their entirety. Advantageously, by constantly monitoring playing media files, CCM 300 can detect undesired activities and enforce those rules defined by CCM 300.

Figure 8 is a diagram of an exemplary high-speed global media content delivery system 20 800, in accordance with one embodiment of the present invention. In one embodiment, system 800 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 210, 220, and/or 230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 800 includes a global delivery network 802 that can include multiple 25 content servers, e.g., 804, 806, 808, 810, 812, 814, and 816, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 804-816 can store a portion, a substantial portion, or the entire contents of a media

content library that can be delivered to client computer systems via a network, e.g., Internet 201, or a WAN (wide area network). Accordingly, each of content server 804-816 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

5

For example, a media delivery point (MDP) 816, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 451, to client computer systems within the Asiatic regions of the world while a media delivery point 812, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 804-816 represents the location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 804-816 can located anywhere within the world, and as such are not limited to the cities shown in global network 802.

15

Still referring to Figure 8, it is further noted that global system 802 is described in conjunction with Figures 2, 3, 4, 5A-D, and 6, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 210 of Figure 2, interacting with a web server, e.g., web server 250 of Figure 2, as described herein, web server 250, in one embodiment, can redirect client computer system 210 to receive the desired media content from an MDP (e.g., 804-816) based on one or more differing criteria.

20

For example, computer system 210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 250 which can be located anywhere in the world. It is noted that steps 702-730 of Figures 7A and 7B can then be performed as described herein such that the

25

present embodiment proceeds to step 732 of Figure 7C. At step 732, the present embodiment can determine which media delivery points, e.g., 804, 806, 808, 810, 812, 814, or 816, can subsequently provide and deliver the desired media content to client computer system 210.

5 Still referring to Figure 8, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 210, e.g., media delivery point 816. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client
10 computer system 210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 804-816) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the
15 embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 210 at step 732, web server 250 transmits to client computer system 210 a redirection command to media delivery point/content server 812 along with a time
20 sensitive access key, also referred to as a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 210 to eventually receive the requested media content. Within system 800, the redirection command can include a time sensitive address of the media content location within media delivery point 812. Accordingly, the New York City media delivery point 812 can subsequently provide and deliver the desired media content to
25 client computer system 210. It is noted that steps 732-742 and step 737 of Figure 7C can be performed by media delivery point 812 in a manner similar to content server 251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 804-816, to provide high fidelity media content to client computer systems, e.g., 210-230, located throughout the world, communication network systems of the Internet 201 do not become overly congested. Additionally, global network 802 can deliver media content to a larger number of client computer systems (e.g., 210-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

10

Referring still to Figure 8, it is noted that media delivery points/content servers 804-816 of global network 802 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 804-816 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that media delivery points 804-816 can be functionally coupled such that if one of them fails, another media delivery point can take over and fulfill its functionality. Additionally, one or more web servers similar to web server 250 can be coupled to global network 802 utilizing wired and/or wireless communication technologies.

15

20

Within system 800, content server/media delivery point 804 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 806-816 of global network 802 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 804.

25

Specifically, the web infrastructure of media delivery point 804 includes firewalls 818 and 820 which are each coupled to global network 802. Firewalls 818 and 820 can be coupled to global network 802 in diverse ways, e.g., utilizing wired and/or wireless communication

technologies. Particularly, firewalls 818 and 820 can each be coupled to global network 702 via a 10/100 Ethernet handoff. However, system 800 is not limited in any fashion to this specific implementation. It is noted that firewalls 818 and 820 are implemented to prevent malicious users from accessing any part of the web infrastructure of media del836, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 840 coupled to device 836
5 while firewall 820 includes a device 838, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 842 coupled to device 838. Furthermore, DB server 840 is coupled with device 838 and DB server 842 is coupled with device 836.

10 Still referring to Figure 8, and within media delivery point 804, firewall 818 is coupled to a director device 822 which is coupled to internal web application server 826 and 828, and a hub server 830. Firewall 820 is coupled to a director 824 which is coupled to internal web application servers 826 and 828, and hub server 830. Hub server 830 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 780 is coupled to
15 a data storage device 832 capable of storing media content. Data storage device 832 can be implemented in a variety of ways, e.g., as a RAID (redundant array of inexpensive/independent disks) appliance.

It is noted that media delivery points 804-816 can be implemented in any manner similar
20 to content server 250 described herein. Additionally, media delivery points 804-816 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 100 of Figure 1.

In another embodiment, CCM 300 can be adapted to be disposed on a media storage
25 device, e.g., media storage device 999 of Figures 10 and 11. Media storage device 999 can be, but is not limited to, a CD, a DVD, or other optical or magnetic storage device. By virtue of disposing a version of CCM 300 on a media storage device 999, embodiments of the present

invention can provide copy protection for audio, video, and other forms of media files that may contain copyrighted material and which may be disposed on a media storage device.

Alternatively, CCM 300 can be adapted to be installed on a computer system, e.g., client computer system 210, via a media storage device 999 upon which it may be disposed.

5

Figure 9 is a block diagram of a copyright compliance mechanism/media storage device (CCM/MSD) 900, a version of CCM 300 adapted to be disposed on a media storage device, e.g., media storage device 999 of Figures 10 and 11. It is noted that CCM 300 in CCM/MSD 900 is analogous to CCM 300 as described in Figures 3, 4, 5A-D, 6A and 7A-C. Further, CCM/MSD 900 can be readily updated in accordance with global delivery system 800, as described in Figures 7A-C, and Figure 8.

In one embodiment, CCM/MSD 900 is adapted to provide stand-alone compliance with copyright restrictions and licensing agreements applicable to media files that may be disposed on a media storage device, e.g., media storage device 999. In another embodiment CCM/MSD 900 is adapted to be installed into a computer system, e.g., client computer system 210 to provide compliance with copyright restrictions and licensing agreements applicable to media files as described in Figures 3, 4, 5A-D, 6A and 7A-C.

Referring to Figure 9, CCM/MSD 900 includes an autorun protocol component 910 for invoking an automatic installation of CCM 300. To deter users from attempts at defeating various features inherent to CCM 300, e.g. the autorun feature, CCM 300's monitoring program, agent program 304, verifies that those features that are to be operational are, and if not, CCM 300 prohibits the user from experiencing of the content of the media storage device.

25

If a user somehow defeats the autorun feature, and the user attempts to utilize an application to capture an image of the content, the application will make an image of the content

on the media storage device, which also images the copyright protection contained thereon, and when the image is played, CCM 300 recognizes the copy protection is present, and CCM 300 will only allow the user to experience the content when authorized, once CCM 300 is installed.

5 By virtue of the protections as described above provided by CCM 300, users will be able to experience the content of the media storage device in the content's original high quality format, thereby obviating the need to compress the media file used on client system 210. Advantageously, the user will no longer need to suffer through poor quality output as a result of severely compressed media files.

10

 It is noted that when adapted to be implemented in conjunction with a secure file format, meaning that the format of the file is, without proper authorization, non-morphogenic, embodiments of the present invention also provide for effective compliance with copyright restrictions and licensing agreements with secure files formats. CCM 300 can control the types
15 of file formats into which the media file can be transformed, e.g., .wav, .mp3, etc.

 In one embodiment, the autorun feature of client system 210 is activated and operational. Alternatively, a notice of required autorun activation may be displayed on the media storage device and/or the case in which the media storage device is stored.

20

 In another embodiment, if CCM 300 is present or if the user is connected to server, then messages containing instructions on how to activate the autorun feature of client system 210 may be presented to the user.

25

 In one embodiment autorun protocol component 910 can detect media storage device drives resident on a computer system, e.g., client computer system 210.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 910 for detecting media storage device drives residing and operable on client computer system 210, in one embodiment of the present invention.

```

5      if ( (dwRetVal = GetLogicalDrives())
        != (DWORD) 0)
    {
        /* initialize variables */
        dwMask = (DWORD) 1;
10      /* initialize path to root of current drive */
        _tcscpy(szDrive, _T("A:\\"));

        for (nIndex = 0, dwMask = (DWORD) 1;
15          dwMask != (DWORD) 0;
            nIndex++, dwMask <<= 1)
        {
            if ((dwRetVal & dwMask) != 0)
            {
20                /* construct path to root of drive */
                szDrive[0] = (TCHAR) 'A' + nIndex;

                if (GetDriveType(szDrive) == DRIVE_CDROM)
                {
25                    MessageBox((HWND) 0,
                                _T("CD-ROM drive found."),
                                szDrive,
                                MB_OK);
                }
            }
            else
            {
30                /* clear bit at current position */
                dwRetVal &= (~dwMask);
            }
        }
35    }
}

```

In another embodiment, autorun protocol component 910 can detect whether a media storage device containing media files has been inserted into a media storage device drive coupled with client computer system 210, e.g., drive 1112. In another embodiment, CCM 300 can include instructions that monitors media storage device drive 1112, and upon detection of drive activation, CCM 300 determines what type of media storage device has been inserted therein.

Subsequently, CCM 300 can detect various triggers on the media storage device to invoke its protection, e.g., a hidden file on newer media storage devices and/or the copyright indicator bit on legacy media storage devices, obviating the need for autorun. Upon detection, CCM 300 can invoke the appropriate protection for the associated media file.

5

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 910 for detecting a media storage device inserted in a media storage device drive residing and operable on client computer system 210, in one embodiment of the present invention.

10

```

    /* set error mode for operation */
    uiErrMode = SetErrorMode(SEM_FAILCRITICALERRORS);

    /* initialize path to root of current drive */
15    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <=<= 1)
20    {
        if ((dwCDROMMask & dwMask) != 0)
        {
            /* construct path to root of drive */
            szDrive[0] = (TCHAR) 'A' + nIndex;
25            if ( GetDiskFreeSpace(szDrive,
                                   &dwSectors,
                                   &dwBytes,
                                   &dwClustersFree,
                                   &dwClusters)
30                != 0)
            {
                /* add bit for drive to mask */
                dwRetVal |= dwMask;
35            }
        }
    }

    /* restore original error mode */
40    SetErrorMode(uiErrMode);

```

Additionally, autorun protocol component 910 can also detect changes in media, e.g., insertion of a different media storage device 999. Further, other media changes can be detected subsequent to adaptation of the source code including, but not limited to, detecting a previously accessed media file, detecting a previously inserted media storage device.

5

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 910 for detecting a change in media, in one embodiment of the present invention.

```

10      /* initialize path to root of current drive */
      _tcscpy(szDrive, _T("A:\\"));
      for (nIndex = 0, dwMask = (DWORD) 1;
15      dwMask != (DWORD) 0;
      nIndex++, dwMask <=<= 1)
      {
          /* check for presence of CD-ROM media in drive */
          if ((dwCurrMask & dwMask) != 0)
          {
20              /* check if media previously in drive */
              if ((dwPrevMask & dwMask) == 0)
              {
                  /* construct path to root of drive */
                  szDrive[0] = (TCHAR) 'A' + nIndex;
25
                  /* check for presence of marker on drive */
                  if (IsMPBMarkerPresent(szDrive) != 0)
                  {
30                      /* process autorun information present on drive */
                      nRetVal = ProcessAutorun(szDrive);
                  }
              }
          }
      }
35

```

Still referring to Figure 9, CCM/MSD 900 also includes a kernel level filter driver 920 for controlling a data input path of an operating system coupled with and operable on client computer system 210.

40

CCM/MSD 900 also includes a generalized filter driver 930 for controlling ripping and burning applications, e.g., Nero, Roxio, Exact Audio Copy, and others, thereby preventing.

The following C++ source code is an exemplary implementation of a portion of
 5 generalized filter driver 930 for controlling ripping and burning applications that may be residing
 on and operable within on client computer system 210, in one embodiment of the present
 invention.

```

10  bool   bDisabled;           /* flag indicating CD reads disabled */
    /* initialize variables */
    bDisabled = false;
    if (bProtected == true)
15  {
        if (type == IRP_MJ_DEVICE_CONTROL)
        {
            ULONG ulIoControlCode = stack-
20  >Parameters.DeviceIoControl.IoControlCode;
            if (ulIoControlCode == IOCTL SCSI_PASS_THROUGH)
            {
                SCSI_PASS_THROUGH * pspt = (SCSI_PASS_THROUGH *)
25  Irp->AssociatedIrp.SystemBuffer;
                if ( (pspt != NULL)
                    && (pspt->Cdb[0] == SCSIOP_READ_CD))
                {
30  pspt->DataTransferLength = 0;
                    pspt->ScsiStatus = 0;
                    bDisabled = true;
                }
            }
35  else if (ulIoControlCode == IOCTL SCSI_PASS_THROUGH_DIRECT)
            {
                SCSI_PASS_THROUGH_DIRECT * psptd =
40  (SCSI_PASS_THROUGH_DIRECT *)
                Irp->AssociatedIrp.SystemBuffer;
                if ( (psptd != NULL)
                    && (psptd->Cdb[0] == SCSIOP_READ_CD))
                {
45  psptd->DataTransferLength = 0;
                    psptd->ScsiStatus = 0;
                }
            }
        }
    }

```

```

        bDisabled = true;
    }
}
5    }
    if (bDisabled == true)
    {
        /* complete current request */
10    status = CompleteRequest(Irp, STATUS_SUCCESS, 0);
    }
    else
    {
        /* pass request down without additional processing */
15    status = IoAcquireRemoveLock(&pdx->RemoveLock, Irp);

        if (!NT_SUCCESS(status))
            return CompleteRequest(Irp, status, 0);

20    IoSkipCurrentIrpStackLocation(Irp);
    status = IoCallDriver(pdx->LowerDeviceObject, Irp);
    IoReleaseRemoveLock(&pdx->RemoveLock, Irp);
    }
25

```

Still referring to Figure 9, CCM/MSD 900 includes a CCM 300, analogous to CCM 300 of Figure 3, and is adapted to be installed in client computer system 210 in the manner described herein.

30

In one embodiment, kernel level filter driver 920, generalized filter driver 930 and CCM 300 of CCM/MSD 900 are automatically installed on client computer system 210, subsequent to insertion of media storage device 999 into a media storage device drive, e.g., media storage device drive 1112 of Figure 11. Autorun protocol component 910, as described above, detects

35 insertion of media storage device 999 into an appropriate drive, and initiates installation of the components, e.g., CCM 300, driver 920 and 930. In one embodiment, drivers 920 and 930 may be temporarily installed and may be deleted upon removal of media storage device 999 from media storage device drive 1112. In another embodiment, drivers 920 and 930 may be installed in hidden directories and/or files within client computer system 210. In another embodiment,

some components of CCM 300 can remain installed on client system 210, e.g. the monitoring program e.g., agent program 304, whereas other components, e.g., the kernel level filter driver 920 can be dynamically loaded and unloaded as necessary, e.g., in accordance with copyright restrictions and licensing agreements applicable to the media file.

5

Embodiments of the present invention utilize software, e.g., CCM/MSD 900, that is placed on media storage device 999, in conjunction with controlling software CCM 300 installed on client computer system 210, and web server 250 and/or content server 251, wherein each component is communicatively coupled with the other via the Internet, thereby enabling dynamic updating of CCM 300 in the manner as described in Figure 4, and steps 716 and 718 of Figures 7A-C.

In the present embodiment, CCM/MSD 900 provides a stand alone DRM that is far more sophisticated than existing DRM solutions. This is because CCM/MSD 900 goes into the data pathway of the operating system operable on client computer system 210 system and obtains control of the data pathway, e.g., filter driver 1108 of Figure 11, rather than exploit inefficiencies or errors in the computer system.

Figure 10 is a block diagram of a communicative environment 1000 for controlling unauthorized reproduction of protected media files disposed on a media storage device. Included in communicative environment 1000 is a media storage device drive 1112 coupled with a client computer system 210 via a data/address bus 110. Client computer system 210 is coupled with web server 250 and content server 251 via Internet 201. A media storage device 999, upon which a CCM/MSD 900 may be disposed, is inserted in media storage device drive 1112. Autorun protocol component 910 detects the insertion and automatically invokes installation of CCM 300, kernel level filter driver 920 and generalized filter driver 930 into client computer system 210. Subsequent to installation, CCM 300 initiates a dynamic update with web server

250 and or content server 251, via Internet 201. By installing CCM 300 on client computer system, agent program 304 (Figure 3) of CCM 300, the monitoring component of CCM 300, can control the integrity of the software and it enables, by conferring with servers 250 and/or 251 via Internet online, the CCM 300 software version on media storage device 999 and installed on
5 client computer system 210 to be updated when circumventions occur and kept current from platform to platform.

Advantageously, CCM 300's agent program 304's monitoring mechanism enables constant morphing of the version of CCM 300 disposed on media storage device 999 by
10 communicating with the server and utilizing the dynamic update capabilities of global network 800 to readily update that which has been installed on client computer system 210, via media storage device 999.

In one embodiment, the installation, clandestine with respect to the user, is initiated by
15 inserting media storage device 999 into an appropriate media storage device drive, e.g. a magnetic/optical disk drive or alternative device drive coupled with client system. If the user is not registered with CCM 300, as described herein with reference to Figure 4 and Figures 7A-7C, once installed, CCM 300 initiates an update process with web server 250 and/or content server 251, to readily include updates that have been invoked subsequent to release of the media file on
20 media storage device 999. By virtue of the dynamic update capabilities of CCM 300, regardless of the version of CCM 300 on media storage device 999, CCM 300 provides compliance with copyright restrictions and licensing agreements applicable to the media file on media storage device 999. Advantageously, enabling dynamic adaptability of CCM 300 provides for continued interoperability with new and updated operating systems, advancements in electronic technology,
25 communication technologies and protocols, and the like, ensuring CCM's effectiveness into the future.

In another embodiment, if the user is a registered user with global delivery system 800, CCM 300 can detect which version is most current. Accordingly, when the version existing on client system 210 is more current than the version (for install) on media storage device 999, CCM 300 can bypass the install process and present the contents contained on media storage device 5 999 to the user for them to experience.

Further advantageous, this technology is backward compatible with media storage device drives manufactured subsequent to 1982. Additionally, CCM 300 is compatible with media storage devices having a copyright indicator bit disposed thereon. The copyright indicator bit 10 has been included on all CDs release since 1982.

In this embodiment, the media file is not encrypted on media storage device 999. In one embodiment, if the media file is encrypted on the computer, it can be decrypted on the computer. However, home players and/or stand alone media playing devices rarely include a decryption 15 mechanism, and to experience the music on a home machine, the music is conventionally not encrypted.

In one embodiment, an additional component of CCM 300 is that the trigger for agent program 304 may be the copyright bit indicator. This means when the copyright indicator bit is 20 detected by CCM 300, the functions of CCM 300 are initiated. Alternatively, in another embodiment, when the copyright bit indicator is not detected, CCM 300 may remain in an un-invoked or idle state. If CCM 300 can detect the copyright bit indicator, CCM 300 can provide the appropriate compliance with regard to copyright restrictions and licensing agreements applicable to the media files.

25

In an alternative embodiment, a trigger control, in the table of contents of a media storage device 999, includes instructions for triggering autorun protocol 910 of CCM/MSD 900 and can

utilize the copyright indicator bit or alternative implementation to trigger the technology. In this manner, CCM 300 can control copyrighted works while public domain material can be experienced and reproduced at a user's discretion. Because autorun is problematic for media storage device manufacturers, embodiments of CCM/MSD 900 can include alternative autorun programs to perform analogous to autorun.

In another embodiment, CCM 300 can invoke its own proprietary player, e.g., custom media device 310 as described in Figure 3, thus enabling increased control of copyright restrictions and licensing agreements applicable to the media file. By invoking custom media device 310, CCM 300 enables user experience of the media file while providing protection against unauthorized reproduction of the media file.

In an alternative embodiment, the media files and the CCM/MSD 900 disposed on a media storage device 999 are encrypted. This implementation is particularly advantageous for demo versions of media files, beta test versions, and the like that may be disposed on media storage device 999. It is noted that the present embodiment is operable in an online environment, meaning that client computer system 210 needs to be communicatively coupled with web server 250 and/or content server 251 to enable a user experience of the content on a demo version of media storage device 999. In this implementation, CCM 300 allows for specific plays for specific users., which can be controlled via network and server.

In another embodiment, CCM 300 can be implemented for demo and pre-release protection. In this embodiment, CCM 300 utilizes sophisticated encryption technology to encrypted the table of contents, and CCM 300, with an associated decrypted key located on client computer system. Encrypting CCM 300 can also deter nefarious attempts to reverse engineer CCM 300. Decryption can be performed using an associated decryption key.

Alternatively, decryption can be performed by a proprietary or custom media player application resident on the demo media storage device.

5 The content of the media storage device is encrypted, using various levels of encryption to provide protection levels commensurate with copyright holders desires and required protection.

For example, a media storage device is delivered to a user or critic for the purposes of review, .The user inserts the media storage device into the appropriate storage device reader or connector coupled with the journalist's computer, and CCM 300 is installed into client system 200 is a
10 manner clandestine to the user. Once installed, CCM 300 initiates a communication session with web server 250/content server 251, where content server 251 would provide authorization for the user to experience the media file on media storage device 999.

Accordingly, if the user, to whom the demo media storage device had been released, had
15 the demo media storage device stolen, or if the user allowed alternative parties try to experience the content of the media storage device, the unauthorized party would have to try to crack the encryption keys and the encryption of the actual content of the media storage device, consuming non-trivial amounts of time.

20 Thus, CCM 300 is able to control which users receive authorization to experience the media file, how many times the user may experience the media file, and CCM 300 may also define a period of time until the media file may no longer be accessible. This may enable copyright holders to release the content on an authorized media storage device prior to pirated copies flooding the market.

25

Accordingly, a demo media storage device 999 may be configured such that user one may get a copy, user four may get a copy, and if it is known that user four will share the demo

with users eight and nine, then the known users would be enabled to experience the media file. Advantageously, by virtue of defining which users can access and experience the media file, any unauthorized sharing of the media file by one of the authorized users can be readily detected, and further sharing or experiencing of the media file may be halted. Additionally, because the
5 authorized user shared the media file in an unauthorized manner, in a worse case scenario, criminal charges could be filed against that user.

It is noted that placing CCM/MSD 900 on a media storage device so as to enable installation of CCM 300 on client system 210 is one manner in which CCM 300 can be installed
10 on client system 210. An alternative manner in which CCM 300 can be installed on client computer system 210 is through cross-pollination. For example, web casters broadcast the media file to the user. The media file has a CCM 300 coupled, and upon downloading onto client computer system 210, embodiments of the present invention enable the installation of CCM 300 onto client computer system 210. In another manner, CCM 300 is incorporated into and
15 becomes part of an operating system operational on client system 200. Alternatively, laws are passed that mandate the inclusion of CCM 300 on each client computer system 210.

Figure 11 is an exemplary logic/bit path block diagram 1100 of a client computer system, e.g., 210, configured with a copyright compliance mechanism 300 for preventing unauthorized
20 reproduction of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 300 is, in one embodiment, coupled with and operational on client system 210 in the manner described with reference to Figures 4, 5A-5D, 6A, and 7A-7C, 9, and 10.

25 Diagram 1100 of Figure 11 includes a media storage device media extraction/creation application 1102 communicatively coupled to operating system input/output subsystem 1104 via wave in line 1121 and wave out line 1138. Operating system input/output subsystem 1104 is

coupled with media storage device class driver 1106 via wave in line 1123 and wave out line 1136. Media storage device class driver 1106 is coupled with filter driver 1108 via wave in line 1125 and wave out line 1134. Filter driver 1108 is coupled with media storage device port driver 1110 via wave in line 1127 and wave out line 1132. Filter driver 1108 is shown to include a switch 1111, controlled by CCM 300 via connection 1160. Media storage device port driver 1110 is coupled with media storage device drive 1112 via wave line in 1129 and wave line out 1130. Media storage device 999, shown to include CMM/MSD 900 is insertable into media storage device drive 1112. Additionally, CCM 300 is coupled with operating system input/output subsystem 1104 via wave in line 1150 and wave out line 1151.

10

In one embodiment, CCM 300 is coupled to and controls a selectable switch 1111 filter driver 1108. Depending upon the copyright restrictions and licensing agreements applicable to a media file disposed on media storage device 999, CCM 300 controls whether switch 1111 is open (shown), thus preventing the media file from reaching media extraction/creation application 1102, or closed (not shown) so as to allow reproduction of a protected media file. Media extraction/creation application 1102 can be a ripping or burning application such as Nero, Roxio, Exact Audio Copy, or other readily available application.

15

Continuing with Figure 11, a media storage device 999 is received by media storage device drive 1112. Agent program 304 of CCM 300 determines whether media storage device 999 or a media file disposed thereon is protected by any copyright restrictions and/or licensing agreements, e.g., via detection of a copyright indicator bit. CCM 300 communicates with filter driver 1108 to control switch 1111 accordingly. In the present example, reproducing media storage device 999, and/or the contents thereon, would violate applicable restrictions and/or agreements and therefore switch 1111 is in an open position, such that the output path to media extraction/creation application 1102, e.g., wave-out line 1138, is effectively blocked, thereby preventing unauthorized reproduction of media storage device 999.

20
25

It is particularly noted that by virtue of CCM 300 controlling switches 1111, and therefore controlling wave-out line 1138, any incoming copyright protected media files disposed
5 on a media storage device 999 can be prevented from being reproduced in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Advantageously, as new secure or proprietary file formats are developed, CCM 300 can
10 be readily adapted to be functional therewith. Further, CCM/MSD 900 can prevent users from unauthorized reproduction of media files, recording, copying, ripping, burning, etc. By using kernel level filter drivers, , e.g., filter driver 1108, and getting low enough into the OS on client system 210 where CCM 300 can detect particular applications and when those applications request media storage device to poll the media file for copying, ripping, etc, CCM 300 can
15 disable the data input path. CCM 300 in this embodiment, deals with the input pathway.

In one embodiment, alternative applications that monitor the state of client computer system 210 can enable the autorun functionality of client computer system 210 or alternatively, invoke an automatic mechanism similar to autorun to ensure invocation of CCM 300 for
20 compliance of copyright restrictions and licensing agreements applicable to media storage device 999 and/or the copyright protected media files disposed thereon

In one embodiment, CCM 300 can invoke a proprietary media player from media storage device 999, or activate a proprietary media player resident and operable on client computer
25 system 210.

When media storage device 999 is a multisession device e.g., a compact disk having a data session and a music session (audio tracks), and it is inserted into media storage device drive 1112, CCM 300 looks at the contents of the media storage device 999, wherein in some operating systems the audio tracks will not be displayed. Instead, the data session is shown, as is
5 an autorun file, e.g., autorun protocol component 910, and upon clicking, invokes a player application. CCM 300 can have a data session and files to which a user may not have access unless a player application is invoked.

In one embodiment, the player application could deposit CCM 300 monitoring portion,
10 e.g., agent program 304 on client system 210, which in one embodiment may reside on client computer system 210 subsequent to removal of media storage device from media storage device drive

By virtue of content in a multisession media storage device 999, which may not be
15 directly accessible to most player applications, at some point the player application will be invoked which could then install the CCM 300 into client system 210, in one embodiment of the present invention.

In one embodiment, a proprietary media player application is on media storage device
20 999. However, it is not automatically invoked. Upon some user intervention, e.g., inserting media storage device 999 into media storage device drive 1112, a media player application is loaded onto client system 210 which has CCM 300 integrated therewith. Thus, CCM 300 is launched regardless of autorun being activated or not activated, and mandates the user to utilize the proprietary media player application to experience the content of the media files on the media
25 storage device, 999.

In an alternative embodiment, client computer system 210 has autorun off, wherein is it common for the user to be unable to play a media file unless a proprietary media player application is invoked. Double clicking to activate the proprietary media player application can initiate an installation of those components of CCM 300 that are bypassed when autorun is not active.

Advantageously, by providing a copyright compliance mechanism, e.g., 300, which can be easily and readily installed in a client computer system, e.g., 210, embodiments of the present invention can be implemented to control access to, control the delivery of, and control the user's experience with media content subject to copyright restrictions and licensing agreements, for example, as defined by the DMCA. Additionally, by closely associating a client computer system, e.g., 210, with the user thereof, and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

A process of installing a copyright compliance mechanism from a media storage device upon which protected media is disposed, in accordance with one embodiment of the present invention can be shown with a flow chart.

A method of preventing unauthorized reproduction of protected media disposed on a media storage device according to one embodiment is described.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 1: METHOD AND SYSTEM FOR PROVIDING USER SELECTABLE AND LOCATION-OBSCURED STREAMING MEDIA

Although previous embodiments of the present invention specifically recite the use of audio, they are also well suited to use video under a similar system such that the source of the video content may not be directly accessed thereby preventing unauthorized retrieval and/or copying of the video content. Furthermore, embodiments of the present invention allow video streaming to be done without proprietary encryption and/or decryption tools or without a proprietary video player.

Additionally, an embodiment of the present invention may include the functionality of periodically changing the media content source address thereby restricting unauthorized users from directly retrieving and/or copying the media content. This may be implemented by utilizing a server network where multiple servers are content providers or by routing a requesting client device through multiple servers. Alternatively, the delivery of media content from a central content provider may be routed through one or more intermediate servers before being received by the requesting client device.

The present embodiment may periodically switch the media delivery route in some way so even if someone with inappropriate intentions is able to find out where the media content source is, the next time they try to use that route (or link) it does not exist anymore. Exemplary code for implementing this type of functionality follows and is labeled "Mstream Program". Additionally, exemplary code for implementing this type of functionality is also labeled "Redirector Program" herein. This type of functionality may be referred to as a directory name change where the present embodiment periodically changes the directory thereby making older routing (or linking) information useless. Within an embodiment of the present invention, the links (or routes) that are actively coupled to one or more client devices are maintained while future links (or routes) are being created for new client devices. It is appreciated that as client devices are uncoupled from the

media content source, that directory link may be immediately changed thereby restricting access to unauthorized client devices.

Another embodiment in accordance with the present invention includes implementing access keys for subscription to media content. The access keys may be used to validate the proper delivery of media content via the media content delivery system described herein. The access keys system of the present embodiment may be implemented in a wide variety of ways in accordance with the present invention. For example, Figure 1-4 is a block diagram of an exemplary method and system for implementing access keys in accordance with an embodiment of the present invention. Specifically, a client device (e.g., computer) may communicatively couple with a web/application server of the present embodiment in order to request a media play list. In response, the web server determines whether the client is authorized to receive the media play list associated with the request. If so, the web server returns an access key to the client device which is appended to the media play list. A media player application operating on the client device sends a request for one or more pieces of media content along with the access key. The application type is then checked by the web server along with the user access key to determine if it is valid. If valid, the web server issues to the client device a redirect command to the current location of the desired media content along with the current valid key (e.g., of that day, hour, etc.). In response, the media player application operating on the client-device makes a new request (along with the current valid key) to a content server for the one or more pieces of media content. The content server validates the current valid key and then retrieves the desired media content from a database and then transmits it to the client device for use by its media player. It is noted that an exemplary implementation of this access key method in accordance with the present embodiment is shown within the "Redirector Program" code contained herein. It is understood that this access key method and system in accordance with the present embodiment may be implemented in conjunction with the media content delivery system described herein.

In yet another embodiment, the present invention provides a rate control restrictor functionality that may be resident in a content server in order to monitor and limit "suspicious" media content requests. For example, if a client device tries to retrieve media content faster than a predefined limit, the rate control restrictor uncouples the client device from the media content source and continues to restrict its access from the source for a predefined amount of time (e.g., 10 minutes). One of the reasons for implementing this rate control restrictor is to restrict access to those unauthorized users that are trying to retrieve and/or copy media content from the source in an unauthorized manner. For example, a client device may be retrieving pieces of audio content at a rate determined to be much faster than is humanly possible to listen to. As such, the rate control restrictor (which may be implemented with software and/or hardware) uncouples the client device from the media content source and restricts its access for a predefined amount of time.

An embodiment of the present invention may be implemented such that a user of a client device may establish a huge buffer of media content (e.g., audio clips, video clips, and the like), however, the present embodiment just provides the client device one piece of media content at a time as would be typically needed to utilize (e.g., listen to, watch, etc.) that piece of media content in real time.

In another embodiment, the present invention may implement embedded keys and/or digital watermarks within media content that may be delivered utilizing one or more of the media content delivery systems described herein. By using embedded keys and/or digital watermarks within media content, it is easier to determine if some unauthorized person has been retrieving and/or copying media content from a media content source. The embedded keys and/or digital watermarks within media content may include, but are not limited to, information indicating where the media came from, the identity of the media requestor and/or the identity of the requestor client device. With this information, it may be

determined by the present embodiment who or what client device has been unauthorized in retrieving and/or copying media content from one or more of the media sources. As such, that person and/or client device can be permanently restricted by the present embodiment from requesting and/or accessing media content.

Figure 1-5 is a block diagram of an embodiment of an exemplary computer system 1-5000 used in accordance with the present invention. It is understood that system 1-5000 is not strictly limited to be a computer system. As such, system 1-5000 of the present embodiment is well suited to be any type of computing device (e.g., server computer, desktop computer, laptop computer, portable computing device, etc.).

Within the discussions of the present invention, certain processes and steps are discussed that are realized, in one embodiment, as a series of instructions (e.g., software program) that reside within computer readable memory units of computer system 1-5000 and executed by a processor(s) of system 1-5000. When executed, the instructions cause computer 1-5000 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 1-5000 of Figure 1-5 comprises an address/data bus 1-5010 for communicating information, one or more central processors 1-5002 coupled with bus 1-5010 for processing information and instructions. Central processor unit(s) 1-5002 may be a microprocessor or any other type of processor. The computer 1-5000 also includes data storage features such as a computer usable volatile memory unit 1-5004, e.g., random access memory (RAM), static RAM, dynamic RAM, etc., coupled with bus 1-5010 for storing information and instructions for central processor(s) 1-5002, a computer usable non-volatile memory unit 1-5006, e.g., read only memory (ROM), programmable ROM, flash memory, erasable programmable read only memory (EPROM), electrically

erasable programmable read only memory (EEPROM), etc., coupled with bus 1-5010 for storing static information and instructions for processor(s) 1-5002.

System 1-5000 also includes one or more signal generating and receiving devices 1-5008 coupled with bus 1-5010 for enabling system 1-5000 to interface with other electronic devices. The communication interface(s) 1-5008 of the present embodiment may include wired and/or wireless communication technology. For example, in one embodiment of the present invention, the communication interface 1-5008 is a serial communication port, but could also alternatively be any of a number of well known communication standards and protocols, e.g., a Universal Serial Bus (USB), an Ethernet adapter, a FireWire (IEEE 1394) interface, a parallel port, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In one embodiment a digital subscriber line (hereinafter DSL) connection may be employed. In such a case the communication interface(s) 1-5008 may include a DSL modem. In any case, the communication interface(s) 1-5008 may provide a communication interface to the Internet.

Optionally, computer system 1-5000 can include an alphanumeric input device 1-5014 including alphanumeric and function keys coupled to the bus 1-5010 for communicating information and command selections to the central processor(s) 1-5002. The computer 1-5000 can include an optional cursor control or cursor directing device 1-5016 coupled to the bus 1-5010 for communicating user input information and command selections to the central processor(s) 1-5002. The cursor directing device 1-5016 can be implemented using a number of well known devices such as a mouse, a track ball, a track pad, an optical tracking device, a touch screen, etc. Alternatively, it is appreciated that a cursor can be directed and/or activated via input from alphanumeric input device 1-5014 using special keys and key sequence commands. The present embodiment is also well

suited to directing a cursor by other means such as, for example, voice commands.

The system 1-5000 of Figure 1-5 can also include a computer usable mass data storage device 1-5018 such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) coupled with bus 1-5010 for storing information and instructions. An optional display device 1-5012 is coupled to bus 1-5010 of system 1-5000 for displaying video and/or graphics. It should be appreciated that optional display device 1-5012 may be a cathode ray tube (CRT), flat panel liquid crystal display (LCD), field emission display (FED), plasma display or any other display device suitable for displaying video and/or graphic images and alphanumeric characters recognizable to a user.

The following page provides a guide that corresponds to flow chart 1-3000 of Figure 1-3 along with the exemplary "Mstream Program" code that follows it. Specifically, the following page provides a brief explanation of flow chart 1-3000 in conjunction with delineating the lines of "Mstream Program" code that correspond with each step of flow chart 1-3000.

Refer to mstream document "patent41.txt"
Specific lines of code refer to flow chart sections.

Flow chart:

Section A:

general http login page
sends user to a verify page data page.
<http://www.themomi.org/login.php>

Section B:

Look up user name do password check from a database.
Check for client cookie. If not found retrieve client id from database
and write new cookie with id.

Section C:

Content retrieval pages:
http://www.themomi.org/museum/goldlist_platinum.html
http://www.themomi.org/museum/be_the_dj_frames_platinum.html
http://www.themomi.org/museum/random_frames_platinum.html
http://www.themomi.org/perl/hi_genre.cgi
etc.

Section D:

Add user id to http request of the form:
#EXTINF:140,The Chantays - Pipeline
http://www.themomi.org:8087/The_Big_Kahuna/The_Chantays_-_Pipeline.mp3?id=123454321

Section E:

Client media request is directed to mid-tier application found at specified port number
In this example: application is listening at port 8087
#EXTINF:140,The Chantays - Pipeline
http://www.themomi.org:8087/The_Big_Kahuna/The_Chantays_-_Pipeline.mp3?id=123454321

Section F:

Examine user application type and handle accordingly:
lines 189 - 295

Section G:

Handle bad requests, known violators,
lines 136-177
lines 274-285

Section H:

Check if valid user:
lines 180 - 187
lines 350 - 374

Section I:

Get current location of constantly moving content from output of "change_dirs" program:
lines 50-55
Redirect to current location:
lines 246-272

```
#!/usr/bin/perl -w

#-----
#                               Mstream Program
#-----
# This program obscures the proper address of the mp3 source file
# and records the requested mp3 file name, requestor ip number, file size
# and date of request.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald (tedfitz@mindspring.com), ICS CREATIVE
# Date: Aug 20, 2001
# Created for: "themomi.org"
# version 4.1 check db authorization
#-----

my $version = 4.1;

#-----

$SIG{CHLD} = "IGNORE";
use IO::Socket::INET;
use DBI();

my $debug = 0; #debugging messages on/off
##### localize these variables #####
# log_name: what name and where you want the log (/path/name.ext)
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
# choices are: changing all the time
# real_dest_url: the name of the web server that serves the mp3's
# send_away_url: location that browsers are sent to if they try to access mp3s.
# short_name: the name of the program
#####
my $log_name = "./logs/log_hi_redir.csv";
my $apache_log = "./logs/access_hi.log";
my $error_log = "./logs/error_hi.log";
my $debug_log = "debug.csv";
my $server_port = 8087;
my $real_dest_url = "www.mpb.tv";
#shoutcast support
my $shout_cast_dir = "content";

my $short_name = "schi_redir.cgi";

require "dir_list"; # read names set by dir changing program to find current destination

my $mp3_root_dir = "";
# change content directory names to prevent direct linking
#-----
if($short_name == "/schi_redir.cgi/i") { $mp3_root_dir = $platinum;}
if($short_name == "/mid_redir.cgi/i") { $mp3_root_dir = $gold;}
if($short_name == "/low_redir.cgi/i") { $mp3_root_dir = $silver;}
if($short_name == "/x_low_redir.cgi/i") { $mp3_root_dir = $bronze;}
#-----
#print "using $mp3_root_dir as content dir\n";
#-----

my $send_away_url = "www.themomi.org";
my $program_name = "http://www.themomi.org/redirs/schi_redir.cgi";

#####

print "\nStarted program: $short_name Version: $version at Port: $server_port \n";

#-----

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                                Type      => SOCK_STREAM,
                                Reuse     => 1,
                                Listen    => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

#####

my $sok = 1; # general;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0; # this is the winamp player
my $real = 0; # this is the real player
my $ripper = 0; # this is for stream ripper etc.

#-----
# Now do main loop. fork off child when connection made
#-----

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here

```

```

(my $port, my $iaddr) = unpack_sockaddr_in($client_info);
my $remote_host_ip = inet_ntoa($iaddr);
my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
if(! defined($remote_hostname) ){ $remote_hostname = "unknown";}
if($debug){
    print "\n ----- \n";
    print "clients ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . " \n";
}
# The first line is Method ( GET or HEAD (if quicktime player)) + url + http version
my $what = "";
my $whole_request = "";
my $bad_monkey = 0;
my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

$date = localtime;

$what = <$client>;

if(! defined($what)){
    print "$short_name encounterd a client who said nothing! $remote_host_ip , $date<-- \n ";
    open (BADMONKEY, ">>$debug_log");
    $bad_monkey = 1;
    print BADMONKEY "$short_name,$remote_host_ip,$remote_hostname,$date\n";
    exit(1);
}

$first_line = $what; #save for error reporinting

if($debug){
    print "\n First line is \" , $what , \" \n";
}

if ($what !~ /^(GET|HEAD)\s+/i) {
    print " sending bad client request message HTTP/1.1 400, no match on GET|HEAD, said:->$what<--\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if ($what == /$mp3_root_dir/i) {
    print "Sent bad request message HTTP/1.1 400, \"$mp3_dir\" name in request\n said:->$what<--\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if ($what == /^(([A-Z]+)\s+([S]+)\s+HTTP\/([d.]+)\s+\/){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;

    # print "song request $what_song\n";
    $what_song =- /\?(.*)/;
    $user_id = $1;
    # print "\tuser id = $user_id\n";
}

}else{
    print " sending bad client HTTP/1.1 400, Bad Request message\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if($remote_host_ip == /12.40.175.194/){
    print $client "HTTP/1.0 302 Found \r\n";
    print $client "Server: Loser-Dumper/3.141592653 \r\n"; #lie to 'em to keep 'em guessing.
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    #print " sending bad client HTTP/1.1 400, Bad Request message\n";
    #print $client "HTTP/1.1 400, \"Bad Request\"";
    print "banned loser $remote_host_ip tried again sent to ring of hell\n";
    exit(1);
}

# print "method is > $what_method< song is >$what_song< version is >$what_version< \n";

if ($what_method !~ /^(GET|HEAD)/i) {
    exit(1);
}

# now look 'em up and check if they are a valid user or not.
my ($status,$access) = validate($user_id);
# print "\nmy function returned a status of $status with an access level of $access\n";
if($status eq "invalid" || $access eq "none"){
    print "\n Sending bad client request message HTTP/1.1 400, invalid user_id \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

```

```

last if(! defined($request));
$request =~ s/{\r\n}//g;
if($debug) {

    print "< $counter >The client said: " . $request . "<\n";

}
#----- WEB BROWSER INFO ----- #
if($request =~ /Mozilla/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Opera/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Omniweb/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /iCab/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- MP3 RIPPER INFO ----- #
if($request =~ /StreamRipper/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- No get for Wget ----- #
if($request =~ /Wget/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- FlashGet fizzles ----- #
if($request =~ /FlashGet/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- GetRight is Wrong ----- #
if($request =~ /GetRight/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- Step on Spiders ----- #
if($request =~ /asterias/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- Web access = No access ----- #
if($request =~ /Web access/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- MP3 CLIENT INFO ----- #
if($request =~ /Windows-Media-Player/io) { $ok = 0; $nsplayer = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /NSPlayer/io) { $ok = 0; $nsplayer = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /RMA/io) { $ok = 0; $real = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Winamp/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /QuickTime/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Xaudio/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /AppleApp/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /iTunes/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Sonique/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /xams/io) { $ok = 1; $nostream = 0; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

=== hold for error reportin ===
if($request =~ /User Agent/io) { $u_a_line = $request; }

last if ($counter > 20 ); #we don't want some client babbling on and on
$counter++;

}#end while

if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
}
print $client "HTTP/$what_version 302 Found \r\n";
#print " I told him HTTP/$what_version 302 Found \r\n";
print $client "Date: $date \r\n";
print $client "Server: IIS/4.2.1 \r\n";
print $client "Keep-Alive: timeout=15, max=100\r\n";

if($ua eq "") { $ok = 0; $nsplayer = 0; $real = 0; $nostream = 0; }
#----- Good guys ----- #
if($real){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    # print "nsplayer or $real player\n";
}
if($nsplayer){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    #print $client "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    #print "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    # print "nsplayer\n";
}
#=== send to shoutcast === #
if($ok){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print $client "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    #print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print "Ok = other player that doesn't need to be told what port to go to\n";
}
if($nostream){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($smozzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if($ripper){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if($ua eq ""){

```

```

    }
    Sok = 1;
    Snsplayer = 0;
    Smozzy = 0;
    Sreal = 0;

    print $client "\r\n\r\n";
    # print "\ndone\n";
    # print ".....\n\n";

    close $client;

    Swhat_song =- /\/(.+)\/(.+).mp3/io;
    Swhat_list = $1;
    Ssong_name = $2;

    #trash filter .....
    if(! defined($what_list)){
        print "no list $remote_host_ip\n"; $what_list = "none\n";
    }
    if(! defined($song_name)){
        print "no song_name $remote_host_ip\n"; $song_name = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "no hostname $remote_host_ip\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($ua)){
        print "no ua\n"; $ua = "none";
        $bad = 1;
    }
    }

    #APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%i(Referer)i\" \"%i(User-agent)i\")
    #Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

    $date =- /\s(.+)\s(.+)\s(.+)\s(.+)/io;
    $day = "$2";
    if ($day < 10){ $day = "0$day"; } #hack to add leading zero

    my $apache_date = "[ $day/$1/$4:$3 -0500]";
    $whole_request =- s/\r//;
    $whole_request =- s/\n//;

    open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
    $a_str = "$remote_host_ip - - $apache_date \" $whole_request\" 302 2 \"$program_name\" \"$ua\"";
    print APACHE_LOG "$a_str\n";
    close APACHE_LOG;

    if($bad || $ua eq "unknown"){
        $first_line =- s/[\r\n]//g;
        open (ERROR_LOG, ">>$error_log");
        my $e_str = "$remote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
        print ERROR_LOG "$e_str\n";
        close ERROR_LOG;
    }#end if error

    close STDOUT;
    close STDERR;
    close STDIN;
    exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
close ($server);

sub validate{
    my $user_id = shift;
    my $invalid = "invalid";
    my $bad_access = "none";
    my $status;
    my $access;
    my $dbh = DBI->connect("DBI:mysql:database=mpb;host=localhost",
        "somename", "somepassword",
        {'RaiseError' => 1}) or die "can't connect to db";
    my $sql = "select status,access from users where id_num = \"$user_id\" ";
    my $sth = $dbh->prepare($sql);
    $sth->execute();
    while (my $ref = $sth->fetchrow_hashref()) {
        $status=$ref->{'status'}; $access = $ref->{'access'};
    }
    if(! defined($status)){
        return $invalid, $bad_access;
    }
    return ($status,$access);
    $dbh->disconnect();
}

#end
.;

```

```
#!/usr/bin/perl

#####
#
# a simple script to sync the content dirs and redir programs
# this script should be called by the kill.pl program
# this script then calls the "go" script to restart redirs
#####
require "dir_list";
require "new_dir_list";

my $song_root = "/usr/local/apache/sites/mpb.tv/htdocs";

my $high = $platinum;
my $mid = $gold;
my $low = $silver;
my $xlow = $bronze;
my $newhigh = $newplatinum;
my $newmid = $newgold;
my $newlow = $newsilver;
my $newxlow = $newbronze;
my $defhigh = 0;
my $defmid = 0;
my $deflow = 0;
my $defxlow = 0;

if(defined($high) && defined($newhigh)){
    print "\nmv $song_root/$high $song_root/$newhigh\n";
    mv $song_root/$high $song_root/$newhigh;
    print "ln -s $song_root/$newhigh $song_root/$high\n";
    ln -s $song_root/$newhigh $song_root/$high;
    $defhigh = 1;
}
if(defined($mid) && defined($newmid)){
    print "\nmv $song_root/$mid $song_root/$newmid\n";
    mv $song_root/$mid $song_root/$newmid;
    print "ln -s $song_root/$newmid $song_root/$mid\n";
    ln -s $song_root/$newmid $song_root/$mid;
    $defmid = 1;
}
if(defined($low) && defined($newlow)){
    print "\nmv $song_root/$low $song_root/$newlow\n";
    mv $song_root/$low $song_root/$newlow;
    print "ln -s $song_root/$newlow $song_root/$low\n";
    ln -s $song_root/$newlow $song_root/$low;
    $deflow = 1;
}
if(defined($xlow) && defined($newxlow)){
    print "\nmv $song_root/$xlow $song_root/$newxlow\n";
    mv $song_root/$xlow $song_root/$newxlow;
    print "ln -s $song_root/$newxlow $song_root/$xlow\n";
    ln -s $song_root/$newxlow $song_root/$xlow;
    $defxlow = 1;
}

#now call kill & restart mstream scripts
./kill.pk'; # just in case any child apps still running
./go';

# give the slowpokes time to get with the program
print "sleeping\n";
sleep 10;

# now remove old links.
if(defined($high) && defined($newhigh) && ($high ne $newhigh)){
    print "\nrm $song_root/$high\n";
    rm $song_root/$high;
}
if(defined($mid) && defined($newmid) && ($mid ne $newmid)){
    print "rm $song_root/$mid\n";
    rm $song_root/$mid;
}
if(defined($low) && defined($newlow) && ($low ne $newlow)){
    print "rm $song_root/$low\n";
    rm $song_root/$low;
}
if(defined($xlow) && defined($newxlow) && ($xlow ne $newxlow)){
    print "rm $song_root/$xlow\n";
    rm $song_root/$xlow;
}

if ($defhigh && $defmid && $deflow && $defxlow){
    print "cp dir_list dir_list_old\n";
    print "cp new_dir_list dir_list\n";
    cp dir_list dir_list_old;
    cp new_dir_list dir_list;
}

```

Operation of Redirection Script

Purpose:

To prevent users from seeing actual location (URL) of the "Music at The MoMI" Encoded music files, by obscuring this location, via redirection, to a hidden or alternate location.

Background:

Typically, a site that hosts MPEG-3 (MP3) encoded music files hosts a link to a specific location on that hosts server which the astute user could utilize to acquire the files for perpetual personal use. This "unauthorized" use is then defrauding the artist of the royalties, opens the hosting company (i.e. The MoMI) to potential lawsuits, and prevents the tracking of usage, payment of the aforementioned royalties, and ascertaining the popularity of the content at the MoMI site.

Prior methods:

In the past, specific methods have been used to reduce the risks identified above.

One method is to add to the encoded file a layer of additional encryption that may only be played on a specified player and allows a single play for a single fee. Additional license fees may be assessed to unlock the music for perpetual use. This method of delivery is used by a company known as "Liquid Audio". (<http://www.liquidaudio.com>) which offers a free player online. While it is true that this is a secure delivery method, the downside is that the users must download a specialized player for the files to their machine or install additional plug-in applications to handle these files.

The newest versions of this player *will* play content other than the Liquid audio format, the fact remains that it is not cross platform compatible with Mac, PC, and Linux users alike. This is a serious limitation in marketing and availability.

A similar approach is that used at MP3.com (<http://www.mp3.com>), which also uses a specialized player. This player integrates a "Buy this CD" button and uses a similar approach to locking the material. The newest versions of this player *will not* play content other than the MP3.com audio format, and as before, the fact remains that it is not cross platform compatible with Mac, PC, and Linux users alike. This is a serious limitation in marketing and availability in this approach as well.

Real Media Corporation has the RealPlayer[™] and RealPlayer Plus[™] players to deliver content which is NOT compliant with MPEG-3 format files, but uses a proprietary encoding and delivery mechanism. The basic player is free, and the upgraded player (Plus version) is Regular Priced at \$29.99. The quality (or lack thereof) is a topic of much discussion on the internet, and along with poor compatibility across platforms (they support Mac and PC only) limits the audience.

Long term security issues exist with Real Media files, as they are streamed to the users' temporary internet files directory and may be recovered from that directory to be hacked.

Lastly, Apple Computer Corporation (<http://www.apple.com>) has a media player which supports the QuickTime™ format of audio, and now supports MPEG-3 format files. The Quicktime™ available players for Mac and PC formats, but no Linux support

Quicktime™ format has already been hacked and unlocked, and is no longer touted as a secure delivery medium, and mediocre sound quality has plagued this format since it's introduction. Apple has recently conceded in this fight and has introduced a player called iTunes player (<http://www.apple.com/itunes/download/index.html>) which is purely MPEG-3 based. This player offers no security enhancements.

The preferred embodiment used at The MoMI is to not encode the data stream differently, or require specialized players. Industry standard players for PC (Windows™ Media Player™), Mac (iTunes™ Player or Quicktime™ 5.0), or Linux (XMMS Player) are available at no cost and are supported by organizations that are perpetual.

The selection of individual songs or play lists create a specialized format of the playlist, which while being fully recognizable to the MP3 player of choice, obscures the actual location of the music from that user by providing a redirection to a hidden location. This is accomplished via a Common Gateway Interface (CGI) program written in Perl Script (attached as appendix 1-I), that receives the users' request, filters the necessary information and requests the real location from the server location, not the users machine. The files are delivered in a streaming fashion, and never reside on the users' machine.

Additionally, this script records the users IP address, song requested, and transfer size. This allows The MoMI to accurately meter royalty payments, clock usage and transfers, and keep a log of song list popularity.

This combination will reduce the risks identified above, provide compensation where due, and prevent theft and unauthorized redistribution of music enjoyed while visiting at The MoMI .

Appendix 1-I

```
#!/usr/bin/perl -w
```

```
#-----
#           Redirector Program
#-----
# This program obscures the proper address of the mp3 source file for due diligence
# purposes and records the requested mp3 file name, requestor ip number, file size
# and date of request to allow for accurate royalty payments & bandwidth monitoring,
# The resulting data is stored in a .csv file for ease of subsequent analysis.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald (tedfitz@mindspring.com)
# Date: Aug 20, 2001
# Created for: "themomi.org"
# version 1.0 Proof of concept.
# version 2.0 rewrite to include a tcp server to get lower level details and
# improve control over output.
# version 2.1 created hi and lo bandwidth versions.
# version 3.0 converted to forking server version for increased client load
# some code portions from "advanced perl programming"
# version 3.1 Added some code to verify the request is a proper GET HEAD type
# or dump 'em. Added some code to prevent runaway child babble by
# limiting client input to 20 lines.
# version 3.2 Changed code to point to new domain and dir structure
# version 3.3 Added remote host ip hostname & user_agent tracking
# version 3.4 Changed exception handling
# version 3.5 Changed handling of children and changed usage log to apache format
# to allow access stats to be recorded by the analog monitoring program
# version 3.6 Added handler to block mp3 players with stream ripping software
# version 3.7 Added more browser exclusions and changes for new destination names
# to prevent direct linking. Change this to automated procedure next.
# version 3.8 minor change to date & block dl managers
# version 3.9 minor change to create error log
# version 4.0 Added internal rate controls via database & debug on/off
# from command line
#----- #
```

```
my $version = 4.0;
```

```
#-----
```

```
$SIG{CHLD} = "IGNORE"; #kill zombies! Oooo scary scary!;
use IO::Socket::INET;
use DBI();
```

```
my $debug = 0; #debugging messages on/off
```

```
##### localize these variables #####
#
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
# choices are: changing all the time
# real_dest_url: location of the content server
```

```

# send_away_url: location that browsers are sent to if they try to access mp3s.
#
#
#program name
my $level = "hi";          # hi , mid, lo, x_lo;
my $short_name = $level . "_redir.cgi";
my $program_name = "http://www.themomi.org/redirs/$short_name";

# logs
my $apache_log = "/usr/local/apache/logs/access_$level.log";
my $error_log = "/usr/local/apache/logs/error_$level.log";

#port
my $server_port = 8081;

#urls
my $real_dest_url = "www.mpb.tv";
my $send_away_url = "www.themomi.org";

#db connection
my $db_host = "AAA.BBB.CCC.DD";
my $db_user = "my user name";
my $db_pw = "some password";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "$db_user", "$db_pw",
                      {'RaiseError' => 1}) or die "can't connect to db";

#rate control songs/min limit
my $limit = 10;

# Changeable access parameters read in from database
my $mp3_root_dir;
my $pass_key;

# change content directory names to prevent direct linking

my $sql_db = "Select dir from file_location ";
$sql_db .= "where band like \"$level\"";
my $sth_db = $dbh->prepare($sql_db);
$sth_db->execute();
my $ref_db = $sth_db->fetchrow_hashref();

$mp3_root_dir = $ref_db->{'dir'};
$sth_db->finish();

if(!defined($mp3_root_dir)) {
print "oops!\n";
#-----
# $mp3_root_dir = "hi_band";
# $mp3_root_dir = "Fatty";
# $mp3_root_dir = "ookii";
#-----

```

```

# $mp3_root_dir = "mid_band";
# $mp3_root_dir = "chu";
# $mp3_root_dir = "norman";
#-----
# $mp3_root_dir = "lo_band";
# $mp3_root_dir = "chisai";
# $mp3_root_dir = "Slim_Jim";
#-----
# $mp3_root_dir = "x_lo_band";
# $mp3_root_dir = "chibi";
# $mp3_root_dir = "Ally_McBeal";
#-----

} #end if

my $sql_key = "Select cache_key from cache_key ";
my $sth_key = $dbh->prepare($sql_key);
    $sth_key->execute();
my $ref_key = $sth_key->fetchrow_hashref();

    $pass_key = $ref_key->{'cache_key'};
    $sth_key->finish();

#####
my $msg = "debug mode off";
while ($_ = $ARGV[0]) {
    shift;
    #print "found $_ \n";
    last if /^--$/;
    if (/^-D/i || /^-X/i || /^-V/i) {
        $msg = "debug mode on";
        $debug = 1;
    }
}
} #end while

print "\nStarted program: $short_name Version: $version at Port: $server_port $msg \n";
print " Using $real_dest_url/$mp3_root_dir as music source dir\n";

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                               Type    => SOCK_STREAM,
                               Reuse   => 1,
                               Listen  => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

#####

# initialize variables used for player discrimination
my $ok = 1;    # on by default;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0; # this is the winamp player
my $real = 0; # this is the real player
my $ripper = 0; # this is for stream ripper etc.
my $speeder = 0; # this is for speeders
my $what = "";
my $whole_request = "";

```

```

my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

# ----- #
#   Now do main loop. fork off child when connection made
# ----- #

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here

        my $client_info = getpeername($client);
        (my $port, my $iaddr) = unpack_sockaddr_in($client_info);
        my $remote_host_ip = inet_ntoa($iaddr);
        my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
        my $date = localtime;

        if(! defined($remote_hostname)) { $remote_hostname = "unknown";}
        if($remote_host_ip =~ /10.10.0.252/o || $remote_host_ip =~ /10.10.0.251/o){
            exit(1);
        }

        if($debug){
            print "\n ----- \n";
            print "Sshort_name: client ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . " \n";
        }

        #evaluate the first line
        $what = <$client>;
        #ignore network health check from x.x.x..104
        if(! defined($what) && $remote_host_ip !~ /209.10.35.104/){
            print "Sshort_name@$server_port encountered a client who said nothing! $remote_host_ip , $date<-- \n "
;
            exit(1);
        }
        $first_line = $what;

        if($debug){ print "\n First line is \"", $what , "\" \n"; }
        if($what !~ /^(GET|HEAD)\s+/i) { #valid requests must start with GET or HEAD
            if($remote_host_ip !~ /209.10.35.104/o){
                print "Sshort_name -> bad req HTTP/1.1 400, no GET HEAD, $remote_host_ip said:->$what<- \n";
            }
            print $client "HTTP/1.1 400, \"Bad Request\"";
            exit(1);
        }
    }
}

```

```

if ($what =~ /$mp3_root_dir/i) {
    print "$short_name -> bad req HTTP/1.1 400, mp3_dir in request, $remote_host_ip said:->$what<- \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

#banned ip list

if ($remote_host_ip =~ /132.239.12.77/
    || $remote_host_ip =~ /66.122.240.11/
    || $remote_host_ip =~ /63.17.233.236/ ) {
    #print " $short_name sending HTTP/1.1 400, banned $remote_host_ip \n said:->$what<- \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

#now parse line, extract what we need or dump 'em
if ($what =~ /^(([A-Z]+)\s+([\S]+)\s+HTTPV([\d.]+\s*)/){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;
} else {
    print " $short_name: sending bad client HTTP/1.1 400, Bad Request message\n";
    print " $short_name: bad req was: $what\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

# print "method is > $what_method< song is >$what_song< .version is >$what_version< \n";

if ($what_method !~ /(GET|HEAD)/i) {
    exit(1);
}

#now loop to get other info lines
while( ($request = <$client>) ne ("r\n" || "r\n\r\n") ){
    last if(! defined($request));
    $request =~ s/[r\n]//g;
    if($debug) {
        print "<$level: $counter>The client said: " . $request . "<\n";
    }
    #----- WEB BROWSER INFO ----- #
    if($request =~ /Mozilla/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /Opera/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /Omniweb/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /iCab/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    #----- MP3 RIPPER INFO ----- #
    if($request =~ /StreamRipper/io) {

```

```

        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- No get for Wget -----#
    if($request =~ /Wget/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- FlashGet fizzles -----#
    if($request =~ /FlashGet/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- GetRight is Wrong -----#
    if($request =~ /GetRight/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- Step on Spiders -----#
    if($request =~ /asterias/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- Web access = No access -----#
    if($request =~ /Web access/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }

    #----- Monica gets the bone -----#
    if($request =~ /monica/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- man what the hell is this? ;-)- -----#
    if($request =~ /Media Jukebox WinInet Reader/io){
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- burn Nero Player -----#
    if($request =~ /NeroMediaPlayer/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- ISMUSIC is deaf -----#
    if($request =~ /ISMUSIC/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }

    #----- I know who Anonymizer is -----#
    if($request =~ /Anonymizer/io) {
        Sok = 0; $ripper = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    #----- MP3 CLIENT INFO -----#
    if($request =~ /Windows-Media-Player/io){
        Sok = 0; $nsplayer = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /NSPlayer/io) {
        Sok = 0; $nsplayer = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /RMA/io) {
        Sok = 0; $real = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /Winamp/io) { Sok = 1; $request =~ /\.:\s*(.+)\s*$/; $ua = $1; }
    if($request =~ /QuickTime/io) { Sok = 1; $request =~ /\.:\s*(.+)$/; $ua = $1;}

```

```

if($request =~ /Xaudio/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}
if($request =~ /AppleApp/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}
if($request =~ /iTunes/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}
if($request =~ /Sonique/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}
if($request =~ /xmms/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}
if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; Sua = $1;}

#### hold for error reportin ===#
if($request =~ /User Agent/io) { $u_a_line = $request; }

last if ($counter > 20 ); #we don't want some client babbling on and on
$counter++;

}#end while

if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
}

##### Rate controls to prevent excessive access to content

$remote_host_ip =~ /\.+\.+/;
# my $last_octet_mod = $1;
# $last_octet_mod = $last_octet_mod % 1; #mod by number of tables we want
# my $time_offset = 1;

my $sql0 = "Select count(ip_address) as loser_count from loser ";
$sql0 .= "where ip_address = \"$remote_host_ip\" ";
$sql0 .= "AND user_agent = \"$ua\"";
my $sth0 = $dbh->prepare($sql0);
$sth0->execute();
my $ref0 = $sth0->fetchrow_hashref();
my $loser_count = $ref0->{'loser_count'};
$sth0->finish();

##
if($loser_count <= 0 ){

my $sql = "SELECT COUNT(request_time) as number_songs_reqs ";
#$sql .= "MINUTE(MIN(request_time)) - MINUTE(now()) as duration ";
$sql .= "FROM rate_control_0 ";
$sql .= "WHERE ";
$sql .= "ip_address = \"$remote_host_ip\" ";
$sql .= "AND ";
$sql .= "user_agent = \"$ua\" ";
$sql .= "AND ";
$sql .= "request_time > now() - INTERVAL 60 SECOND";

print "\n$sql\n" if $debug;

my $sth = $dbh->prepare($sql);
$sth->execute();

my $ref = $sth->fetchrow_hashref();

```

```

my $song_count = $ref->{'number_songs_reqs'};
#my $duration = ($ref->{'duration'});
    $sth->finish();

print "\n Song count is $song_count \n" if $debug;

if($song_count >= $limit){
    $speeder = 1;
    $ok = 0;
    print "\n Loser exceeded limit\n" if $debug ;

    my $sql1 = "insert into loser (ip_address,user_agent,loser_time) values (\'$remote_host_ip\',\'$ua\',now()
)";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if song_count

}else{
    print "\n loser is in loser table\n" if $debug;
    $speeder = 1 ;
    $ok = 0;
}

if(!$speeder){
    my $sql1 = "INSERT INTO rate_control_0 ";
    $sql1 .= "(ip_address,user_agent,request_time) ";
    $sql1 .= "VALUES (\'$remote_host_ip\',\'$ua\',now())";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if

print $client "HTTP/$what_version 302 Found \r\n";
print " I told him HTTP/$what_version 302 Found \n" if $debug;
print $client "Date: $date \r\n";
print $client "Server: Apache/1.3.20 \r\n"; #lie to 'em to keep 'em guessing.
print $client "Keep-Alive: timeout=10, max=20\r\n";

#----- Good guys ----- #
if(($nsplayer || $real) && ! $speeder){
    print $client "Location: http://$real_dest_url:$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url:$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url:$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    print "found nsplayer or real player\n" if $debug;
}
if($ok && ! $speeder){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($mozy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    # print "mozy type web server being redirected to index page\n";
}

```

```

    }
    if($ripper){
        print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
        # print "mozzy type web server being redirected to index page\r\n";
    }
    if( $speeder ){
        print "Sshort_name: Speeder @ $remote_host_ip sending-bad client HTTP/1.1 400, Bad Request ~
message\r\n";
        print "Sshort_name: Speeder said $what\r\n";
        print $client "HTTP/1.1 400, \"Bad Request\"";
    }

    $ok = 1;
    $nsplayer = 0;
    $smozzy = 0;
    $real = 0;

    print $client "\n\n\r\n";
    # print "\ndone\r\n";
    # print "*****\n\n";

#    $dbh->disconnect();
    close $client;

    $what_song =~ /(.(+)\.(.+).mp3)/io;
    $what_list = $1;
    $song_name = $2;

    #trash filter *****
    if(! defined($what_list)){
        print "Sshort_name: no list\r\n"; $what_list = "none\r\n";
    }
    if(! defined($song_name)){
        print "Sshort_name: no song_name $remote_host_ip\r\n"; $song_name = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "Sshort_name: no hostname\r\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($remote_host_ip)){
        print "Sshort_name: no host_ip\r\n"; $remote_host_ip = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "Sshort_name: no hostname $remote_host_ip\r\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($ua)){
        print "Sshort_name: no ua\r\n"; $ua = "none";
        $bad = 1;
    }

    #match this: APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}\r\n\" \"%{User-agent}\r\n\")
    #change date from: Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

```

```

$date =~ /.+\s(.+)\s(.+)\s(.+)\s(.+)/io;
my $month_name = "$1";
my $day = "$2";
my $time = "$3";
my $year = "$4";

if ($day < 10){ $day = "0$day"; } #hack to add leading zero

$month_name =~ s/ //g;
#my $apache_date = "$day/$1/$4:$3 -0500";
my $apache_date = "$day/$month_name/$year:$time -0500";
$whole_request =~ s/\r//;
$whole_request =~ s/\n//;

open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
$a_str = "Sremote_host_ip - - $apache_date \"$whole_request\" 302 2 \"$program_name\" \"$u_a\"";
print APACHE_LOG "$a_str\n";
close APACHE_LOG;

if($bad || $ua eq "unknown" ){
    $first_line =~ s/[\r\n]//g;
    open (ERROR_LOG, ">>$error_log");
    my $e_str = "Sremote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
    print ERROR_LOG "$e_str\n";
    close ERROR_LOG;
} #end if error

close STDOUT;
close STDERR;
close STDIN;

exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
$dbh->disconnect();
close ($server);
exit(0);

```

Operation of Chooser Script

Purpose:

To provide guests a user selectable variety of the "Music at The MoMI" Encoded music files, by allowing the guest to select individual songs from all of the available lists present at this location for their listening pleasure.

Background:

Typically, a site that hosts music files provides a vast quantity of links that are generically organized by musical style or genre. These are, in turn, links to the files that will be played. Selections may be saved on some sites, but the sites do not have any mechanism for allowing a user based selection in the form of a unique list selected from the overall musical content at that location, but rather the selections are usually single-play choices, and when a saved list approach is used, the user must traverse the entirety of the site to select the musical choices.

Prior Implementations:

MyMP3.com is the closest implementation that could mimic the functionality of the MoMI site. A user may select a low-fi (56K encoded music file) version for saving as a cookie based list for future use. The list is perpetual, meaning that the list is the list until the user edits out the songs that they no longer want on the list. There is only one list saved, and the list must be played from that location while on their site.

Preferred Embodiment:

The musical play lists (lists of songs available) of all varieties are loaded from the MoMI server and passed to a Common Gateway Interface (CGI) program that is written in Perl Script, where the lists are concatenated into a single dimensioned array.

In contrast to prior implementations, the MoMI site provides (via the choose.cgi program) the ability to select songs at whimsy from the available content, and save them for future re-use. Alternately, they can be executed immediately and not saved. A guest of the MoMI site could save hundreds of these customized playlists to their desktop, and play them at will. The advantage here is that the user has a reason to return again and again, based on the depth of the musical selection, to select more variations in listening.

The actual code is provided as appendix 1-II.

Appendix 1-II

```

#!/usr/bin/perl -w
#-----
#           User choice Program
# This program reads in a mp3 play list source file, displays the file names,
# on a web page to allow for selection, and creates a new playlist file to download.
# Programmer: Ted Fitzgerald, ICS CREATIVE
# Initial Creation Date: July 26, 2001
# Created for: "themomi.org"
# version 1.0 proof of concept / prototype / extremely easy to understand code
# version 2.0 port to unix, localize to utilize site dir structure
# version 2.1 rewrite to include customer design changes,
#       display cover.jpg's and m3u file found in each sub dir mp3 directory.
# version 2.2 add fix for malformed .m3u files
# version 3.0 change display format to 2 column mode
# version 4.0 add select all in category button and supporting logic.
# version 4.1 add image for select button and add all supporting logic.
#       We now need to look for param(Submit.x) for a submitted pagea
# version 5.0 Complete rewrite to use database
#-----
$VERSION = "5.0";

$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# ---- declare some variables used in the code program ----- #
# ---- Don't touch these --- #

my $dir_cnt = 0; #counter
my $qry1 = "";
my $qry2 = "";
my $sql = "";

# -----YOU CAN/SHOULD LOCALIZE THESE VARIABLES -----
#

my $rootdir = "mp3";
my $image_path = "http://www.themomi.org/museum/images";
my $image_url = "www.themomi.org/$rootdir";
my $song_root = "../htdocs/";
my $earthc = "themomi.earthc.net";

# ----- Start Code ----- #

$|=1; #flush
print "Content-type: text/html\n\n"; # send basic header
use CGI qw(:standard);

```

```

use CGI::Carp('fatalsToBrowser');
use DBI();
use strict;

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "NOBODY", "ASK_FOR_JOE",
                      {'RaiseError' => 1}) or die "can't connect to db";

my @play_list;
my $artist;
my $song_name;

print<<HTML_DONE;
<html>
<head>
<META NAME="Organization" CONTENT="www.themomi.org">
<META NAME="Author" CONTENT="Ted Fitzgerald">
<META NAME="Quote" CONTENT="My hovercraft is full of eels!">
<META NAME="Description" CONTENT="Music selection / creation software">
<SCRIPT LANGUAGE="JavaScript">
<!--
function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}
// -->
</SCRIPT>
<style><!--a{text-decoration:none}!--></style>
<style><!--a: hover{color:#3399cc; }--></style>

</head>
<body bgcolor="black">
<font size=2 color=white face = arial,san serif, helvetica >
<form method="post">

HTML_DONE

#----- Main page display starts here (selection portion ----- #

# if first time page is displayed submit button hasn't been clicked and submit.blah is
undefined.
# we use this parameter to show initial selection or results portion of page

if(! defined(param("Submit.x"))){ #if first time page is displayed submit is undef

```

```

print<<HTML_DONE;
  <center>
    <a id="button">&nbsp;</a>
    <a name="button">&nbsp;</a>
    <input type="image" value="Submit" src="$image_path/oy.gif" border="0"
alt="Create Playlist" name="Submit">
    <!-- <A HREF="http://www.themomi.org/perl/choose_hi.cgi"
      ONMOUSEOVER="changeImages('oy',
'http://www.themomi.org/museum/images/oy.gif'); return true;"
      ONMOUSEOUT="changeImages('oy',
'http://www.themomi.org/museum/images/oy_over.gif'); return true;">
    <IMG NAME="oy" SRC="http://www.themomi.org/museum/images/oy.gif"
BORDER=0></A> -->
    <br><br>
    <input type=Reset value="Reset" name=Reset>
  </center>
HTML_DONE

# ----- MAIN CODE STARTS HERE ----- #

my $sql = "select distinct playlist_name from m3u";
$sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
  my $playlist_name = $ref->{'playlist_name'};
  push(@play_list,$playlist_name);

  my $seven = $dir_cnt % 2; # flip flop the output to make 2 columns odd = left even =
right

  #make left right columns
  if(($seven)) {
    print "<td align=center valign=top width=50%>";
  } else {
    print "<table align=center border=0 width=90% bordercolor=black name=outer>";
#start outer table
    print "<tr><td align=center valign=top width=50%>";
  } # end if

  print "<br><br><img src=http://$earhc/$image_url/$playlist_name/cover.jpg
align=center><br><br>";

  print "<table border=0 align=center width=90% bordercolor=black
name=internal_table>";

```

```

print "<tr><td><input type=checkbox name=d$dir_cnt></td>";
print "<td><font size=2 color=white face = arial,san serif, helvetica >";
print " --- Select All in Category ---</font>";
print "</td></tr>";

my $sql1 = "select pl_index,artist,song_name from m3u where playlist_name like
\"$playlist_name\"";
# print "$sql1<br>";
my $sth1 = $dbh->prepare($sql1);
$sth1->execute();

while (my $ref1 = $sth1->fetchrow_hashref()) {
    my $id = $ref1->{'pl_index'}; $artist = $ref1->{'artist'}; $song_name = $ref1->{'song_name'};
    $artist =~ s/_//g;
    $song_name =~ s/_//g;
    print "<tr><td><input type=checkbox name=\"$id\"></td>\n";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$ song_name<br></font></td></tr>\n";
} #end while ref1

print "</table>"; # end inner table

if($even) {
    print "</td></tr>";
    print "</table>"; #end outer table
    print "<center><br><font size=2 color=white face = arial,san serif, helvetica >";
    print "<a href=\"\"#button\">Return to top</a></font></center>";
} else {
    print "</td>";
} #end ending left right column if

$dir_cnt++; # increment counter used for left/right flip flop

} #end while ref -> display of play lists loop
#-----
print "<br><br>";

print "<input type=hidden name=size value=$dir_cnt>";

# $dbh->disconnect();

print "</form>";

```

```

} #end if (Submit)
#-- done displaying the selection list --

```

```

#-----#
#----- DISPLAY RESULTS OF SELECTIONS -----#
#
#-----#

```

```

if(defined(param("Submit.x")) || defined(param("Submit.x"))){
  my @play_list;
  my @list_copy;
  my @p_list;
  my $pl;
  my $first_pl;
  my $qry1,
  my $qry2;
  # $num_of_dirs = param("size");

```

```

#

```

```

#my $sql = "select distinct playlist_name from m3u order by playlist_name asc";
my $sql = "select distinct playlist_name from m3u";
  $sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
  my $playlist_name = $ref->{'playlist_name'};
  push(@play_list,$playlist_name);
} #end while "get all playlists" query
#

```

```

#-----#
# parse param list for "all in playlist" request = "d+some num" & individual songs "num"
#-----#
my @song_list;
#my $pl_limit = 5;
#my $song_limit = 50;
#my $pl_count = 1;

foreach (param()) {
  push(@song_list,$_) if /^d+$/;    #just a song number
  # print "<br>$_" if /^d+$/;

```

```

if (/^d\d+$/){          #all in playlist d+number
    s/^d//;             #remove letter "d"

    push(@p_list,$play_list[$_]); #store playlist number
    push(@list_copy,$play_list[$_]);

} # end if "d+numb all in playlist "
} # end foreach "item in param() list"
# -----
#    Build "All in playlist" query string
# -----

if(@p_list){

    my $first_pl = shift @p_list;
    my @list_copy = $first_pl;
    $qry1 = "SELECT * FROM m3u WHERE playlist_name IN ( \"$first_pl\" ";

    foreach my $pl (@p_list){
        # last if($pl_count = $pl_limit);
        #print "<br>=". @p_list; #print "<br>@p_list";

        $qry1 .= ", \"$pl\" ";

        push(@list_copy,$pl);
        # $pl_count++;
    } #end foreach loop to build items in query string

    $qry1 .= ") order by playlist_name"; #    query ends here

    #print "<br>$qry1";

} #end if anything in p_list array

# -----
#    Build individually selected songs query string
# -----
if(@song_list){ #anyone home in song lists? go for it!

    my @song_copy = @song_list;

    my $first = shift @song_list;
    $qry2 = "SELECT * FROM m3u WHERE pl_index IN ( $first" ;
    foreach my $blarg (@song_list){
        #print "<br>=". @song_list;
        #print "<br>@song_list";
    }
}

```

```

    $qry2 .= ", $blarg ";
} #
$qry2 .= ")";
# $qry2 .= "limit $song_limit"; #add limit to number of songs
#print "<br>$qry2";
} #end if anything in song_list array

# -----
#   Now create m3u playlist file
#   Execute query statement(s) and write file
# -----

#print "<br>here's your check of array size greater than zero
defined($qry1),defined($qry2)<br>";
if($qry1 || $qry2){
    # --- Create unique play list name using numbers from date parts ----- #
    my $theDate = localtime;
    $theDate =~ /(\d+):(\d+):(\d+)\b/i;
    my $play_list_name = "Momi_playlist_$2$3.m3u";
    # ----- #

    print "<center><font size=2 color=white face = arial,san serif, helvetica >";
    print "<b>Songs in your play list:</b></font></center>";
    print " <table align=center>";

    open(OUTFILE,">../htdocs/temp/$play_list_name");

    print OUTFILE '#EXTM3U'."\n"; #write file header
    #print '<br>#EXTM3U<br>'; #write file header

    if($qry1){
        my $list = "";
        # now do the playlists
        my $sth1 = $dbh->prepare($qry1);
        $sth1->execute();
        while (my $ref1 = $sth1->fetchrow_hashref()) {
            my $playlist_name = $ref1->{'playlist_name'};
            my $pl_header = $ref1->{'pl_header'};
            my $song_url = $ref1->{'song_url'};
            $song_url =~ s/8080/8081/g;
            print OUTFILE "$pl_header\n$song_url\n";
        } # end while
        #print "<b>my list_copy array = @list_copy</b><br>";
        foreach my $list (@list_copy){
            $list =~ s/_//g;

```

```

        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >All Songs
from <b>\\"$list\\"</b></font></td>
></tr>\n";
    } # end foreach
} #end if $qry1 is defined

if($qry2){
    # now do the selected songs
    my $sth2 = $dbh->prepare($qry2);
    $sth2->execute();
    while (my $ref2 = $sth2->fetchrow_hashref()) {
        my $artist = $ref2->{'artist'};
        my $song_name = $ref2->{'song_name'};
        my $pl_header = $ref2->{'pl_header'};
        my $song_url = $ref2->{'song_url'};
        # $print "$pl_header<br>$song_url<br>" ;
        $artist =~ s/_/_/g;
        $song_name =~ s/_/_/g;
        $song_url =~ s/8080/8081/g;
        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$song_name</font></td></tr>";
        print OUTFILE "$pl_header\n$song_url\n";
    } # end while
} # end if $qry2 is defined

close OUTFILE;

print "</table>";
print "<center><br><br><br>";
print "<a href=\"http://www.themomi.org/temp/$play_list_name\">";
print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
print "</font>";
print "<br><br><br>";

# ----- include text for nubies ----- #

print "<br><br>";
print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the button above to launch
your playlist<br> with your favorite .mp3/.m3u player
</td>

```

```
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>
```

```
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
```

```
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the button above and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
```

```
<tr><td colspan=3 align=center>
MORE
```

```
#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
```

```
print "</table><br><br>";
```

```
}else{
  print "<br><b>Nothing Selected</b><br>";
```

```
  #foreach (param()) {
  #   print "$_<br>";
  #}#end foreach
  #print "The value of Param(\"Submit.x\") is " , param("Submit.x") , "<br>";
  #print "The value of defined Param(\"Submit.x\") is " , defined(param("Submit.x")) ,
  "<br>";
```

```
  $dbh->disconnect();
```

```
}#end if defined queries
}# end if (Submit.x) "display results of selection "
```

```
print "</body>";
print end_html();
```

```
#bye bye! Thanks for visting!
```

```
exit(0);
```

```
l;
```

Operation of Randomizer Script

Purpose:

To provide guests a varied selection of the "Music at The MoMI" Encoded music files, by randomly selecting a group of files for their listening pleasure from all of the available lists present at this location.

Background:

Typically, a site that hosts music files provides a vast quantity of links that are generically organized by musical style or genre. These are, in turn, links to the files that will be played. Selections may be saved on some sites, but the sites do not have any mechanism for allowing a random selection of the overall musical content at that location.

Prior Implementations:

None

Preferred Embodiment:

The musical play lists (lists of songs available) of all varieties are loaded from the MoMI server and passed to a Common Gateway Interface (CGI) program that is written in Perl Script, where the lists are concatenated into a single dimensioned array. Using a random number generator that incorporates as its product the combination of:

- 1) The number produced by the random function (not a truly random number, but random sequence that follows the "seeded" number. This seeded number is a starting point for the further permutations of random selection, but would have been predictable with a fixed seed value.)
- 2) The Julian date in the server including the hours, minutes, seconds, day, date, and four-digit year.

Using that combination, the following code is executed:

```
$theDate = 'date'; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "Momi_random_playlist_$2$3.m3u";

$random_seed = "$3$2$3"; #use time parts to make random
srand $random_seed; # seed the random num gen.
```

As can be seen, the time/date then provides a basis for seeding the random number generator that is only repeatable once per millennium.

Now that a random number has been selected, the actual process of applying this to the array of song titles begins.

The criteria for selecting a song for inclusion in the random play list is three-fold:

- 1) There cannot be more than 50 songs total in the list;
- 2) The selection must come from no more than 20 play lists
- 3) The selection must not have been previously selected.

So assuming that the first two conditions have not been met, the resulting random number is used as the index into the array and will select the song title that resides in that location in the array. If that title has been previously selected, a new number is requested, and the process continues until all 3 conditions are fulfilled.

The actual code is provided as appendix 1-III.

The resulting list of song titles are then formatted into a usable playlist (or M3U file as it is known), and it is written to the server's hard disk while a page is prepared for presenting this to the user.

The code then formats an HTML page for the viewer to see that contains a link to the M3U file that is on the MoMI server, which they can either execute immediately, or download. A downloaded playlist cannot provide information to the user regarding the real location of the files, as the original playlists do not contain that information. (see document on the redirector script).

Appendix 1-III

```

#!/usr/bin/perl -w
#-----
#
#          RANDOM PLAY LIST PROGRAM
#
# This program reads in a mp3 play list source file in many directories ,
# displays the random choice of file names ,
# and creates a new file to download.
# Programmer: Ted Fitzgerald, ICS Creative
# Date: July 26, 2001
# Created for "themomi.org"
# version 1.0 proof of concept
# version 2.0 port to unix , rewrite to include customer design changes
#      display cover.jpg's and m3u file found in each sub dir mp3 directory.
# version 2.1 added more randomness to the choices
# version 3.0 rewrote the code to use the cgi O.O mode for speed improvements.
# version 4.0 rewrote the code to increase the randomness and limit the number of
#      selections that are possible. Added tons of comments.
# version 5.0 rewrote app to use the mysql database, March 17 2002
#
#-----
my $version;
$version = 5.0;
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# declare variables

my ($i,$j); #just counters
my ($play_list_name); #variable to create random file name for download
my ($show_many_songs); # = total number of songs in all the dirs.
my (@thename); #array to contain the name of the randomly selected song
my (@thepath); #array to contain the URL path of the randomly selected song
my ($showname); #used to find the portion of name we want to display.
my ($song_url);
my (@theheader);
my (@artist);
my ($artist);
my ($theDate);
#----- user adjustable paramerters -----
my $max_num_songs = 34; # set limit of how many songs to display
my $maxdirs = 20; #set limit on number of dirs to look in
my $mp3dir = "../htdocs/mp3"; #path from cgi script to mp3 directory
my $center_col_size = ($max_num_songs / 2 + 1);

$|=1; #flush the buffer
use strict;

```

```

use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
    "PIZZA_MAN", "HE_DELIVERS",
    {'RaiseError' => 1}) or die "can't connect to db";

my $q = new CGI;

print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>Random Play List Creation</title>";
print "<style><!--a{text-decoration:none}!--></style>";
print "<style><!--a: hover{color:#3399cc; }--></style>";

print <<HTML_DONE;
<SCRIPT LANGUAGE="JavaScript">
<!--

function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}

// -->
</SCRIPT>
<head>
<body bgcolor="black" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="2" text="white" link="#ffcc66" vlink="#ffcc
66" alink="#ffcc66">

<font size=2 color=white face = arial,san serif, helvetica >
HTML_DONE

$theDate = localtime; #get system date
$theDate =~ /(d+):(d+):(d+)\b/i;
$play_list_name = "Momi_random_playlist_$3$2$3.m3u";
#-----#
# do all file io before displaying the names in list

```

```
my $sql = "select pl_header,song_name,song_url,artist from m3u order by rand() limit
$max_num_songs";
```

```
my $sth = $dbh->prepare($sql);
    $sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $pl_header = $ref->{'pl_header'};
    my $song_name = $ref->{'song_name'};
    my $song_url = $ref->{'song_url'};
    my $artist = $ref->{'artist'};
    push (@thename,$song_name);
    push (@thepath,$song_url);
    push (@theheader,$pl_header);
    push (@artist,$artist);
}
```

```
open(OUTFILE,">./htdocs/temp/$play_list_name");
#write out file header
print OUTFILE "#EXTM3U\n";
for($i=0;$i<$max_num_songs;$i++){
```

```
    print OUTFILE "$theheader[$i]\n";
    $song_url = "$thepath[$i]";
    $song_url =~ s/8080/8081/io;
    print OUTFILE "$song_url\n";
```

```
}#end for
close OUTFILE;
```

```
#-----#
#now show 'em what they have in the list
```

```
print<<MORE;
```

```
<center>
```

```
MORE
```

```
print "<br>";
```

```
print "</center>";
```

```
print "<table align=\"center\" border=0 width = \"95%\">";
```

```
print "<tr width=45%><td><td rowspan=$center_col_size align=center valign=top>";
```

```
print <<HERE;
```

```

<A HREF="http://www.themomi.org/temp/$play_list_name"
    ONMOUSEOVER="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert_over.gif');
return true;"
    ONMOUSEOUT="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert.gif'); return
true;">
    <IMG NAME="create_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/create_now_vert.gif"
BORDER=0></A>
HERE

print "</td>";
print "<td width=45%></td></tr>";
for($i=0;$i<$max_num_songs;$i++){
    my $flip_flop = ($i % 2 );

    $showname = $thename[$i];
    # $showname =~ /EXTINF:\d+,(.+)/i;
    $showname =~ s/_/_/g;
    $artist = $artist[$i];
    $artist =~ s/_/_/g;

    if($flip_flop){
        print "<td width = 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showname
</font></td></tr>";
    } else {
        print "<tr><td width= 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showna
me </font></td>";
    }
}
print "</table><br>";

#-----#

# print "<center>";
# print "<br><a href=\"http://www.themomi.org/temp/$play_list_name\">";
# print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
# print "</center>";
print "<table align=center border=0 width=90%>";
print<<MORE;

```

```

<tr><td width=40% align=center>&nbsp;</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">
MORE

print "<A HREF=\"http://www.themomi.org/perl/random_hi.cgi\"";

print<<MORE;
  ONMOUSEOVER="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm_over.gif');
  return true;"
  ONMOUSEOUT="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm.gif');
  return true;">
    <IMG NAME="rerandom_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/re-randomize_sm.gif"
BORDER=0></A>
</font></td>
<td>&nbsp;</td>
</tr>
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the center button above to launch
your <br>playlist with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the center button and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
<tr><td colspan=3 align=center>
MORE

#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
print "</table><br><br>";
print "<center>";

print $q->end_html;

1;

```

Section 2: SYSTEM AND METHOD FOR PROVIDING USER SELECTABLE AND
LOCATION-OBSCURED STREAMING MEDIA

Although previous embodiments of the present invention specifically recite the use of audio, they are also well suited to use video under a similar system such that the source of the video content may not be directly accessed thereby preventing unauthorized retrieval and/or copying of the video content. Furthermore, embodiments of the present invention allow video streaming to be done without proprietary encryption and/or decryption tools or without a proprietary video player.

Additionally, an embodiment of the present invention may include the functionality of periodically changing the media content source address thereby restricting unauthorized users from directly retrieving and/or copying the media content. This may be implemented by utilizing a server network where multiple servers are content providers or by routing a requesting client device through multiple servers. Alternatively, the delivery of media content from a central content provider may be routed through one or more intermediate servers before being received by the requesting client device.

The present embodiment may periodically switch the media delivery route in some way so even if someone with inappropriate intentions is able to find out where the media content source is, the next time they try to use that route (or link) it does not exist anymore. Exemplary code for implementing this type of functionality follows and is labeled "Mstream Program". Additionally, exemplary code for implementing this type of functionality is also labeled "Redirector Program" herein. This type of functionality may be referred to as a directory name change where the present embodiment periodically changes the directory thereby making older routing (or linking) information useless. Within an embodiment of the present invention, the links (or routes) that are actively coupled to one or more client devices are maintained while future links (or routes) are being created for new client devices. It is appreciated that as client devices are uncoupled from the

media content source, that directory link may be immediately changed thereby restricting access to unauthorized client devices.

Another embodiment in accordance with the present invention includes implementing access keys for subscription to media content. The access keys may be used to validate the proper delivery of media content via the media content delivery system described herein. The access keys system of the present embodiment may be implemented in a wide variety of ways in accordance with the present invention. For example, Figure 2-4 is a block diagram of an exemplary method and system for implementing access keys in accordance with an embodiment of the present invention. Specifically, a client device (e.g., computer) may communicatively couple with a web/application server of the present embodiment in order to request a media play list. In response, the web server determines whether the client is authorized to receive the media play list associated with the request. If so, the web server returns an access key to the client device which is appended to the media play list. A media player application operating on the client device sends a request for one or more pieces of media content along with the access key. The application type is then checked by the web server along with the user access key to determine if it is valid. If valid, the web server issues to the client device a redirect command to the current location of the desired media content along with the current valid key (e.g., of that day, hour, etc.). In response, the media player application operating on the client device makes a new request (along with the current valid key) to a content server for the one or more pieces of media content. The content server validates the current valid key and then retrieves the desired media content from a database and then transmits it to the client device for use by its media player. It is noted that an exemplary implementation of this access key method in accordance with the present embodiment is shown within the "Redirector Program" code contained herein. It is understood that this access key method and system in accordance with the present embodiment may be implemented in conjunction with the media content delivery system described herein.

In yet another embodiment, the present invention provides a rate control restrictor functionality that may be resident in a content server in order to monitor and limit "suspicious" media content requests. For example, if a client device tries to retrieve media content faster than a predefined limit, the rate control restrictor uncouples the client device from the media content source and continues to restrict its access from the source for a predefined amount of time (e.g., 10 minutes). One of the reasons for implementing this rate control restrictor is to restrict access to those unauthorized users that are trying to retrieve and/or copy media content from the source in an unauthorized manner. For example, a client device may be retrieving pieces of audio content at a rate determined to be much faster than is humanly possible to listen to. As such, the rate control restrictor (which may be implemented with software and/or hardware) uncouples the client device from the media content source and restricts its access for a predefined amount of time.

An embodiment of the present invention may be implemented such that a user of a client device may establish a huge buffer of media content (e.g., audio clips, video clips, and the like), however, the present embodiment just provides the client device one piece of media content at a time as would be typically needed to utilize (e.g., listen to, watch, etc.) that piece of media content in real time.

In another embodiment, the present invention may implement embedded keys and/or digital watermarks within media content that may be delivered utilizing one or more of the media content delivery systems described herein. By using embedded keys and/or digital watermarks within media content, it is easier to determine if some unauthorized person has been retrieving and/or copying media content from a media content source. The embedded keys and/or digital watermarks within media content may include, but are not limited to, information indicating where the media came from, the identity of the media requestor and/or the identity of the requestor client device. With this information, it may be

determined by the present embodiment who or what client device has been unauthorized in retrieving and/or copying media content from one or more of the media sources. As such, that person and/or client device can be permanently restricted by the present embodiment from requesting and/or accessing media content.

Figure 2-5 is a block diagram of an embodiment of an exemplary computer system 2-5000 used in accordance with the present invention. It is understood that system 2-5000 is not strictly limited to be a computer system. As such, system 2-5000 of the present embodiment is well suited to be any type of computing device (e.g., server computer, desktop computer, laptop computer, portable computing device, etc.). Within the discussions of the present invention, certain processes and steps are discussed that are realized, in one embodiment, as a series of instructions (e.g., software program) that reside within computer readable memory units of computer system 2-5000 and executed by a processor(s) of system 2-5000. When executed, the instructions cause computer 2-5000 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 2-5000 of Figure 2-5 comprises an address/data bus 2-5010 for communicating information, one or more central processors 2-5002 coupled with bus 2-5010 for processing information and instructions. Central processor unit(s) 2-5002 may be a microprocessor or any other type of processor. The computer 2-5000 also includes data storage features such as a computer usable volatile memory unit 2-5004, e.g., random access memory (RAM), static RAM, dynamic RAM, etc., coupled with bus 2-5010 for storing information and instructions for central processor(s) 2-5002, a computer usable non-volatile memory unit 2-5006, e.g., read only memory (ROM), programmable ROM, flash memory, erasable programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), etc., coupled with bus 2-5010 for storing static information and instructions for processor(s) 2-5002.

System 2-5000 also includes one or more signal generating and receiving devices 2-5008 coupled with bus 2-5010 for enabling system 2-5000 to interface with other electronic devices. The communication interface(s) 2-5008 of the present embodiment may include wired and/or wireless communication technology. For example, in one embodiment of the present invention, the communication interface 2-5008 is a serial communication port, but could also alternatively be any of a number of well known communication standards and protocols, e.g., a Universal Serial Bus (USB), an Ethernet adapter, a FireWire (IEEE 1394) interface, a parallel port, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In one embodiment a digital subscriber line (hereinafter IDSL) connection may be employed. In such a case the communication interface(s) 2-5008 may include a IDSL modem. In any case, the communication interface(s) 2-5008 may provide a communication interface to the Internet.

Optionally, computer system 2-5000 can include an alphanumeric input device 2-5014 including alphanumeric and function keys coupled to the bus 2-5010 for communicating information and command selections to the central processor(s) 2-5002. The computer 2-5000 can include an optional cursor control or cursor directing device 2-5016 coupled to the bus 2-5010 for communicating user input information and command selections to the central processor(s) 2-5002. The cursor directing device 2-5016 can be implemented using a number of well known devices such as a mouse, a track ball, a track pad, an optical tracking device, a touch screen, etc. Alternatively, it is appreciated that a cursor can be directed and/or activated via input from alphanumeric input device 2-5014 using special keys and key sequence commands. The present embodiment is also well suited to directing a cursor by other means such as, for example, voice commands.

The system 2-5000 of Figure 2-5 can also include a computer usable mass data storage device 2-5018 such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) coupled with bus 2-5010 for storing information and instructions. An optional display device 2-5012 is coupled to bus 2-5010 of system 2-5000 for displaying video and/or graphics. It should be appreciated that optional display device 2-5012 may be a cathode ray tube (CRT), flat panel liquid crystal display (LCD), field emission display (FED), plasma display or any other display device suitable for displaying video and/or graphic images and alphanumeric characters recognizable to a user.

The following page provides a guide that corresponds to flow chart 2-3000 of Figure 2-3 along with the exemplary "Mstreem Program" code that follows it. Specifically, the following page provides a brief explanation of flow chart 2-3000 in conjunction with delineating the lines of "Mstreem Program" code that correspond with each step of flow chart 2-3000.

Refer to mstream document "patent41.txt"
Specific lines of code refer to flow chart sections.

Flow chart:

Section A:

general http login page
sends user to a verify page data page.
<http://www.themomi.org/login.php>

Section B:

Look up user name do password check from a database.
Check for cleint cookie. If not found retrieve client id from database
and write new cookie with id.

Section C:

Content retrieval pages:
http://www.themomi.org/museum/goldlist_platinum.html
http://www.themomi.org/museum/be_the_dj_frames_platinum.html
http://www.themomi.org/museum/random_frames_platinum.html
http://www.themomi.org/perl/hi_genre.cgi
etc.

Section D:

Add user id to http request of the form:
#EXTINF:140,The Chantays - Pipeline
http://www.themomi.org:8087/The_Big_Kahuna/The_Chantays_-_Pipeline.mp3?id=123454321

Section E:

Client media request is directed to mid-tier applicaton found at specified port number
In this example: application is listening at port 8087
#EXTINF:140,The Chantays - Pipeline
http://www.themomi.org:8087/The_Big_Kahuna/The_Chantays_-_Pipeline.mp3?id=123454321

Section F:

Examine user application type and handle accourdingly:
lines 189 - 295

Section G:

Handle bad requests, known violators,
lines 136-177
lines 274-285

Section H:

Check if valid user:
lines 180 - 187
lines 350 - 374

Section I:

Get current location of constantly moving content from output of "change_dirs" program:
lines 50-55
Redirect to current location:
lines 246-272

```
#!/usr/bin/perl -w

#-----
#                               Mstream Program
#-----
# This program obscures the proper address of the mp3 source file
# and records the requested mp3 file name, requestor ip number, file size
# and date of request.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald (tedfittz@mindspring.com), ICS CREATIVE
# Date: Aug 20, 2001
# Created for: "themomi.org"
# version 4.1 check db authorization
#-----

my $version = 4.1;

#-----

$SIG{CHLD} = "IGNORE";
use IO::Socket::INET;
use DBI();

my $debug = 0; #debugging messages on/off
##### localize these variables #####
# log_name: what name and where you want the log (/path/name.ext)
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
# choices are: changing all the time
# real_dest_url: the name of the web server that serves the mp3's
# send_away_url: location that browsers are sent to if they try to access mp3s.
# short_name: the name of the program
#####
my $log_name = "./logs/log_hi_redir.csv";
my $apache_log = "./logs/access_hi.log";
my $error_log = "./logs/error_hi.log";
my $debug_log = "debug.csv";
my $server_port = 8087;
my $real_dest_url = "www.mpb.tv";
#shoutcast support
my $shout_cast_dir = "content";

my $short_name = "schi_redir.cgi";

require "dir_list"; # read names set by dir changing program to find current destination

my $mp3_root_dir = "";
# change content directory names to prevent direct linking
#-----
if($short_name =~ /schi_redir.cgi/i) { $mp3_root_dir = $platinum;}
if($short_name =~ /mid_redir.cgi/i) { $mp3_root_dir = $gold;}
if($short_name =~ /low_redir.cgi/i) { $mp3_root_dir = $silver;}
if($short_name =~ /x_low_redir.cgi/i){ $mp3_root_dir = $bronze;}

#-----
#print "using $mp3_root_dir as content dir\n";
#-----

my $send_away_url = "www.themomi.org";
my $program_name = "http://www.themomi.org/redirs/schi_redir.cgi";

#####

print "\nStarted program: $short_name Version: $version at Port: $server_port \n";

#-----

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                                Type => SOCK_STREAM,
                                Reuse => 1,
                                Listen => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

#####

my $ok = 1; # general;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0; # this is the winamp player
my $real = 0; # this is the real player
my $ripper = 0; # this is for stream ripper etc.

#-----
# Now do main loop. fork off child when connection made
#-----

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here
        my $client_info = getpeername($client);
```

```

(my Sport, my $iaddr) = unpack_sockaddr_in($client_info);
my $remote_host_ip = inet_ntoa($iaddr);
my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
if(! defined($remote_hostname)) { $remote_hostname = "unknown";}
if($debug){
    print "\n ----- \n";
    print "clients ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . " \n";
}
# The first line is Method ( GET or HEAD (if clicktime player)) + url + http version
my $what = "";
my $whole_request = "";
my $bad_monkey = 0;
my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

$date = localtime;

$what = <$client>;

if(! defined($what)){
    print "$short_name encountered a client who said nothing! $remote_host_ip , $date-- \n ";
    open (BADMONKEY, ">>$debug_log");
    $bad_monkey = 1;
    print BADMONKEY "$short_name, $remote_host_ip, $remote_hostname, $date\n";
    exit(1);
}

$first_line = $what; #save for error reporting

if($debug){
    print "\n First line is \"$what\" , \"$first_line\" \n";
}

if ($what !~ /^(GET|HEAD)\s+/i) {
    print "sending bad client request message HTTP/1.1 400, no match on GET|HEAD, said:-->$what<--\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if ($what =~ /$mp3_root_dir/i) {
    print "Sent bad request message HTTP/1.1 400, \"$mp3_dir\" name in request\n said:-->$what<--\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if ($what =~ /^([A-Z])\s+([S])\s+HTTP\/([d\.-])\s+/i){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;

    # print "song request $what_song\n";
    $what_song =~ /\?(\.+)/;
    $user_id = $1;
    # print "\tuser id = $user_id\n";
}

else{
    print "sending bad client HTTP/1.1 400, Bad Request message\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

if($remote_host_ip =~ /12.40.175.194/){
    print $client "HTTP/1.0 302 Found \r\n";
    print $client "Server: Loser-Dumper/3.141592653 \r\n"; #lie to 'em to keep 'em guessing.
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    #print "sending bad client HTTP/1.1 400, Bad Request message\n";
    #print $client "HTTP/1.1 400, \"Bad Request\"";
    print "banned loser $remote_host_ip tried again sent to ring of hell\n";
    exit(1);
}

# print "method is > $what_method< song is >$what_song< version is >$what_version< \n";

if ($what_method !~ /^(GET|HEAD)/i) {
    exit(1);
}

# now look 'em up and check if they are a valid user or not.
my ($status, $success) = validate($user_id);
# print "\nmy function returned a status of $status with an access level of $success\n";
if($status eq "invalid" || $success eq "none"){
    print "\n Sending bad client request message HTTP/1.1 400, invalid user_id \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

while( ($request = <$client>) ne (" \r\n" || "\r\n\r\n") ){

```

```

last if(! defined($request));
$request =~ s/{\r\n}//g;
if($debug) {
    print "< Scounter >The client said: " . $request . "<\n";
}
#----- WEB BROWSER INFO ----- #
if($request =~ /Mozilla/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Opera/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Omniweb/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /iCab/io) { $ok = 0; $smozzy = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- MP3 RIPPER INFO ----- #
if($request =~ /StreamRipper/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- No get for Wget ----- #
if($request =~ /Wget/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- FlashGet fizzles ----- #
if($request =~ /FlashGet/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- GetRight is Wrong ----- #
if($request =~ /GetRight/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- Step on Spiders ----- #
if($request =~ /asterias/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- Web access = No access ----- #
if($request =~ /Web access/io) { $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#----- MP3 CLIENT INFO ----- #
if($request =~ /Windows-Media-Player/io) { $ok = 0; $nsplayer = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /NSPlayer/io) { $ok = 0; $nsplayer = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /RMA/io) { $ok = 0; $real = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Winamp/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /QuickTime/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Xaudio/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /AppleApp/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /iTunes/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /Sonique/io) { $ok = 0; $nostream = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /xams/io) { $ok = 1; $nostream = 0; $request =~ /\.+:\s*(.+)/; $ua = $1; }
if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)/; $ua = $1; }

#=== hold for error reportin ===#
if($request =~ /User Agent/io) { $u_a_line = $request; }

last if ($counter > 20 ); #we don't want some client babbling on and on
$counter++;

}#end while

if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
}

print $client "HTTP/$what_version 302 Found \r\n";
#print " I told him HTTP/$what_version 302 Found \r\n";
print $client "Date: $date \r\n";
print $client "Server: IIS/4.2.1 \r\n";
print $client "Keep-Alive: timeout=15, max=100\r\n";

if($ua eq "") { $ok = 0; $nsplayer = 0; $real = 0; $nostream = 0; }
#----- Good guys ----- #
if($real){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    # print "nsplayer or $real player\n";
}
if($nsplayer){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    #print $client "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    #print "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    # print "nsplayer\n";
}
#=== send to shoutcast === #
if($ok){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print $client "Location: http://$real_dest_url:8000/$shout_cast_dir$what_song\r\n";
    #print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print "Ok = other player that doesn't need to be told what port to go to\n";
}
if($nostream){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($smozzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if($ripper){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if($ua eq ""){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
}

```

```

    }
    $ok = 1;
    $nsplayer = 0;
    $smozzy = 0;
    $sreal = 0;

    print $client "\r\n\r\n";
    # print "\ndone\n";
    # print "*****\n\n";

    close $client;

    $what_song = - /\/(.+)\/(.+).mp3/io;
    $what_list = $1;
    $song_name = $2;

    #trash filter *****
    if(! defined($what_list)){
        print "no list $remote_host_ip\n"; $what_list = "none\n";
    }
    if(! defined($song_name)){
        print "no song_name $remote_host_ip\n"; $song_name = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "no hostname $remote_host_ip\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($ua)){
        print "no ua\n"; $ua = "none";
        $bad = 1;
    }
    }

    #APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\")
    #Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

    $date = - ./.\s(.+)\s(.+)\s(.+)\s(.+)/io;
    $day = "$2";
    if ($day < 10){ $day = "0$day"; } #hack to add leading zero

    my $apache_date = "[ $day/$1/$4:$3 -0500]";
    $whole_request = - s/\r//;
    $whole_request = - s/\n//;

    open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
    $a_str = "$remote_host_ip - - $apache_date \"$whole_request\" 302 2 \"$program_name\" \"$ua\"";
    print APACHE_LOG "$a_str\n";
    close APACHE_LOG;

    if($bad || $ua eq "unknown"){
        $first_line = - s/[\r\n]//g;
        open (ERROR_LOG, ">>$error_log");
        my $e_str = "$remote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
        print ERROR_LOG "$e_str\n";
        close ERROR_LOG;
    }#end if error

    close STDOUT;
    close STDERR;
    close STDIN;
    exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
close ($server);

sub validate{
    my $user_id = shift;
    my $invalid = "invalid";
    my $bad_access = "none";
    my $status;
    my $access;
    my $dbh = DBI->connect("DBI:mysql:database=mpb;host=localhost",
        "somename", "somepassword",
        {'RaiseError' => 1}) or die "can't connect to db";
    my $sql = "select status,access from users where id_num = \"$user_id\" ";
    my $sth = $dbh->prepare($sql);
    $sth->execute();
    while (my $ref = $sth->fetchrow_hashref()) {
        $status=$ref->('status'); $access = $ref->('access');
    }
    if(! defined($status)){
        return $invalid, $bad_access;
    }
    return ($status,$access);
    $dbh->disconnect();
}#end
1;

```

```
#!/usr/bin/perl

#####
#
# a simple script to sync the content dirs and redir programs
# this script should be called by the kill.pl program
# this script then calls the "go" script to restart redirs
#####
require "dir_list";
require "new_dir_list";

my $song_root = "/usr/local/apache/sites/mpb.tv/htdocs";

my $high = $platinum;
my $mid = $gold;
my $low = $silver;
my $xlow = $bronze;
my $newhigh = $newplatinum;
my $newmid = $newgold;
my $newlow = $newsilver;
my $newxlow = $newbronze;
my $defhigh = 0;
my $defmid = 0;
my $deflow = 0;
my $defxlow = 0;

if(defined($high) && defined($newhigh)){
    print "\nmv $song_root/$high $song_root/$newhigh\n";
    mv $song_root/$high $song_root/$newhigh;
    print "ln -s $song_root/$newhigh $song_root/$high\n";
    ln -s $song_root/$newhigh $song_root/$high;
    $defhigh = 1;
}
if(defined($mid) && defined($newmid)){
    print "\nmv $song_root/$mid $song_root/$newmid\n";
    mv $song_root/$mid $song_root/$newmid;
    print "ln -s $song_root/$newmid $song_root/$mid\n";
    ln -s $song_root/$newmid $song_root/$mid;
    $defmid = 1;
}
if(defined($low) && defined($newlow)){
    print "\nmv $song_root/$low $song_root/$newlow\n";
    mv $song_root/$low $song_root/$newlow;
    print "ln -s $song_root/$newlow $song_root/$low\n";
    ln -s $song_root/$newlow $song_root/$low;
    $deflow = 1;
}
if(defined($xlow) && defined($newxlow)){
    print "\nmv $song_root/$xlow $song_root/$newxlow\n";
    mv $song_root/$xlow $song_root/$newxlow;
    print "ln -s $song_root/$newxlow $song_root/$xlow\n";
    ln -s $song_root/$newxlow $song_root/$xlow;
    $defxlow = 1;
}

#now call kill & restart mstream scripts
./kill.pk; # just in case any child apps still running
./go;

# give the slowpokes time to get with the program
print "sleeping\n";
sleep 10;

# now remove old links.
if(defined($high) && defined($newhigh) && ($high ne $newhigh)){
    print "\nrm $song_root/$high\n";
    rm $song_root/$high;
}
if(defined($mid) && defined($newmid) && ($mid ne $newmid)){
    print "rm $song_root/$mid\n";
    rm $song_root/$mid;
}
if(defined($low) && defined($newlow) && ($low ne $newlow)){
    print "rm $song_root/$low\n";
    rm $song_root/$low;
}
if(defined($xlow) && defined($newxlow) && ($xlow ne $newxlow)){
    print "rm $song_root/$xlow\n";
    rm $song_root/$xlow;
}

if ($defhigh && $defmid && $deflow && $defxlow){
    print "cp dir_list dir_list_old\n";
    print "cp new_dir_list dir_list\n";
    cp dir_list dir_list_old;
    cp new_dir_list dir_list;
}

```

FIGURE 2-6 is a diagram of an exemplary global media content delivery system 2-6000 in accordance with an embodiment of the present invention. System 2-6000 may be utilized to globally deliver media content (e.g., audio, video, etc.) to one or more client devices in any manner similar to that described herein. System 2-6000 includes a global media content delivery network 2-6004 along with one or more web infrastructures 2-6002. The global media content delivery network 2-6004 is communicatively coupled with one or more web infrastructures 2-6002. It is understood that the web infrastructure 2-6002 and the global media content delivery network 2-6004 may be coupled in a wide variety of ways in accordance with the present embodiment. For example, the web infrastructure 2-6002 and the global media content delivery network 2-6004 may be coupled utilizing wired and/or wireless communication technologies.

The global media content delivery network 2-6004 may include multiple "points of presence" that may be located throughout the world. It is understood that each point of presence may be implemented in a manner similar to web infrastructure 2-6002. Within Figure 2-6, these points of presence are represented by small circles along with the exemplary names Tokyo, London, Chicago and Amsterdam. It is appreciated that the web infrastructure 2-6002 is a fully redundant system architecture. Furthermore, each web infrastructure 2-6002 point of presence of the global media content delivery network 2-6004 may store a substantial portion or the entire portion of a media content library that may be provided to client devices via a network, such as, the Internet or a wide area network (WAN). Additionally, all of the web infrastructure 2-6002 points of presence of the global media content delivery network 2-6004 may be interconnected. As -such, if one of the web infrastructure 2-6002 points of presence fails for some reason, another point of presence may take over and fulfill its functionality.

By having these web infrastructure 2-6002 points of presence, each one is able to provide media content to client devices in the vicinity of the world where it is

located. For example, the Tokyo point of presence is able to provide media content from its media content library to client devices within the Asian part of the world while the Chicago point of presence is able to provide media content from its media content library to client devices within the United States and Canada. It is understood that each point of presence of the global media content delivery network 2-6004 may be utilizing the exemplary "Mstreem Program" code along with one or more of the other codes and/or embodiments described herein in order to deliver media content.

Operation of Redirection Script

Purpose:

To prevent users from seeing actual location (URL) of the "Music at The MoMI" Encoded music files, by obscuring this location, via redirection, to a hidden or alternate location.

Background:

Typically, a site that hosts MPEG-3 (MP3) encoded music files hosts a link to a specific location on that hosts server which the astute user could utilize to acquire the files for perpetual personal use. This "unauthorized" use is then defrauding the artist of the royalties, opens the hosting company (i.e. The MoMI) to potential lawsuits, and prevents the tracking of usage, payment of the aforementioned royalties, and ascertaining the popularity of the content at the MoMI site.

Prior methods:

In the past, specific methods have been used to reduce the risks identified above.

One method is to add to the encoded file a layer of additional encryption that may only be played on a specified player and allows a single play for a single fee. Additional license fees may be assessed to unlock the music for perpetual use. This method of delivery is used by a company known as "Liquid Audio". (<http://www.liquidaudio.com>) which offers a free player online. While it is true that this is a secure delivery method, the downside is that the users must download a specialized player for the files to their machine or install additional plug-in applications to handle these files.

The newest versions of this player *will* play content other than the Liquid audio format, the fact remains that it is not cross platform compatible with Mac, PC, and Linux users alike. This is a serious limitation in marketing and availability.

A similar approach is that used at MP3.com (<http://www.mp3.com>), which also uses a specialized player. This player integrates a "Buy this CD" button and uses a similar approach to locking the material. The newest versions of this player *will not* play content other than the MP3.com audio format, and as before, the fact remains that it is not cross platform compatible with Mac, PC, and Linux users alike. This is a serious limitation in marketing and availability in this approach as well.

Real Media Corporation has the RealPlayer™ and RealPlayer Plus™ players to deliver content which is NOT compliant with MPEG-3 format files, but uses a proprietary encoding and delivery mechanism. The basic player is free, and the upgraded player (Plus version) is Regular Priced at \$29.99. The quality (or lack thereof) is a topic of much discussion on the internet, and along with poor compatibility across platforms (they support Mac and PC only) limits the audience.

Long term security issues exist with Real Media files, as they are streamed to the users' temporary internet files directory and may be recovered from that directory to be hacked.

Lastly, Apple Computer Corporation (<http://www.apple.com>) has a media player which supports the QuickTime[™] format of audio, and now supports MPEG-3 format files. The Quicktime[™] available players for Mac and PC formats, but no Linux support

Quicktime[™] format has already been hacked and unlocked, and is no longer touted as a secure delivery medium, and mediocre sound quality has plagued this format since it's introduction. Apple has recently conceded in this fight and has introduced a player called iTunes player (<http://www.apple.com/itunes/download/index.html>) which is purely MPEG-3 based. This player offers no security enhancements.

The preferred embodiment used at The MoMI is to not encode the data stream differently, or require specialized players. Industry standard players for PC (Windows[™] Media Player[™]), Mac (iTunes[™] Player or Quicktime[™] 5.0), or Linux (XMMS Player) are available at no cost and are supported by organizations that are perpetual.

The selection of individual songs or play lists create a specialized format of the playlist, which while being fully recognizable to the MP3 player of choice, obscures the actual location of the music from that user by providing a redirection to a hidden location. This is accomplished via a Common Gateway Interface (CGI) program written in Perl Script (attached as appendix 2-I), that receives the users' request, filters the necessary information and requests the real location from the server location, not the users machine. The files are delivered in a streaming fashion, and never reside on the users' machine.

Additionally, this script records the users IP address, song requested, and transfer size. This allows The MoMI to accurately meter royalty payments, clock usage and transfers, and keep a log of song list popularity.

This combination will reduce the risks identified above, provide compensation where due, and prevent theft and unauthorized redistribution of music enjoyed while visiting at The MoMI .

Appendix 2-I

```
#!/usr/bin/perl -w
```

```
#-----
#           Redirector Program
#-----
# This program obscures the proper address of the mp3 source file for due diligence
# purposes and records the requested mp3 file name, requestor ip number, file size
# and date of request to allow for accurate royalty payments & bandwidth monitoring,
# The resulting data is stored in a .csv file for ease of subsequent analysis.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald (tedfitz@mindspring.com)
# Date: Aug 20, 2001
# Created for: "themomi.org"
# version 1.0 Proof of concept.
# version 2.0 rewrite to include a tcp server to get lower level details and
#           improve control over output.
# version 2.1 created hi and lo bandwidth versions.
# version 3.0 converted to forking server version for increased client load
#           some code portions from "advanced perl programming"
# version 3.1 Added some code to verify the request is a proper GET HEAD type
#           or dump 'em. Added some code to prevent runaway child babble by
#           limiting client input to 20 lines.
# version 3.2 Changed code to point to new domain and dir structure
# version 3.3 Added remote host ip hostname & user_agent tracking
# version 3.4 Changed exception handling
# version 3.5 Changed handling of children and changed usage log to apache format
#           to allow access stats to be recorded by the analog monitoring program
# version 3.6 Added handler to block mp3 players with stream ripping software
# version 3.7 Added more browser exclusions and changes for new destination names
#           to prevent direct linking. Change this to automated procedure next.
# version 3.8 minor change to date & block dl managers
# version 3.9 minor change to create error log
# version 4.0 Added internal rate controls via database & debug on/off
#           from command line
#----- #
```

```
my $version = 4.0;
```

```
#-----
$SIG{CHLD} = "IGNORE"; #kill zombies! Oooo scary scary!;
use IO::Socket::INET;
use DBI();
```

```
my $debug = 0; #debugging messages on/off
```

```
##### localize these variables #####
#
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
#           choices are: changing all the time
# real_dest_url: location of the content server
```

```

# send_away_url: location that browsers are sent to if the try to access mp3s.
#
#
#program name
my $level = "hi";          # hi, mid, lo, x_lo;
my $short_name = $level . "_redir.cgi";
my $program_name = "http://www.themomi.org/redirs/$short_name";

# logs
my $apache_log = "/usr/local/apache/logs/access_$level.log";
my $error_log = "/usr/local/apache/logs/error_$level.log";

#port
my $server_port = 8081;

#urls
my $real_dest_url = "www.mpb.tv";
my $send_away_url = "www.themomi.org";

#db connection
my $db_host = "AAA.BBB.CCC.DD";
my $db_user = "my user name";
my $db_pw = "some password";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "$db_user", "$db_pw",
                      {'RaiseError' => 1}) or die "can't connect to db";

#rate control songs/min limit
my $limit = 10;

# Changeable access parameters read in from database
my $mp3_root_dir;
my $pass_key;

# change content directory names to prevent direct linking

my $sql_db = "Select dir from file_location ";
$sql_db .= "where band like \"\$level\"";
my $sth_db = $dbh->prepare($sql_db);
$sth_db->execute();
my $ref_db = $sth_db->fetchrow_hashref();

$mp3_root_dir = $ref_db->{'dir'};
$sth_db->finish();

if(!defined($mp3_root_dir)) {
print "oops!\n";
#-----
# $mp3_root_dir = "hi_band";
# $mp3_root_dir = "Fatty";
# $mp3_root_dir = "ookii";
#-----
}

```

```

# $mp3_root_dir = "mid_band";
# $mp3_root_dir = "chu";
# $mp3_root_dir = "norman";
#-----
# $mp3_root_dir = "lo_band";
# $mp3_root_dir = "chisai";
# $mp3_root_dir = "Slim_Jim";
#-----
# $mp3_root_dir = "x_lo_band";
# $mp3_root_dir = "chibi";
# $mp3_root_dir = "Ally_McBeal";
#-----

} #end if

my $sql_key = "Select cache_key from cache_key ";
my $sth_key = $dbh->prepare($sql_key);
    $sth_key->execute();
my $ref_key = $sth_key->fetchrow_hashref();

$pass_key = $ref_key->{'cache_key'};
$sth_key->finish();

#####
my $msg = "debug mode off";
while ($_ = $ARGV[0]) {
    shift;
    #print "found $_ \n";
    last if /^-$/;
    if (/^-D/i || /^-X/i || /^-V/i) {
        $msg = "debug mode on";
        $debug = 1;
    }
}
} #end while

print "\nStarted program: $short_name Version: $version at Port: $server_port $msg \n";
print " Using $real_dest_url/$mp3_root_dir as music source dir\n";

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                                Type    => SOCK_STREAM,
                                Reuse   => 1,
                                Listen  => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

#####

# initialize variables used for player discrimination
my $ok = 1;    # on by default;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0;  # this is the winamp player
my $real = 0;    # this is the real player
my $ripper = 0;  # this is for stream ripper etc.
my $speeder = 0; # this is for speeders
my $what = "";
my $whole_request = "";

```

```

my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

# ----- #
#   Now do main loop. fork off child when connection made
# ----- #

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here

        my $client_info = getpeername($client);
        (my $port, my $iaddr) = unpack_sockaddr_in($client_info);
        my $remote_host_ip = inet_ntoa($iaddr);
        my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
        my $date = localtime;

        if(! defined($remote_hostname)){ $remote_hostname = "unknown";}
        if($remote_host_ip =~ /10.10.0.252/o || $remote_host_ip =~ /10.10.0.251/o){
            exit(1);
        }

        if($debug){
            print "\n ----- \n";
            print "$short_name: client ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . " \n";
        }

        #evaluate the first line
        $what = <$client>;
        #ignore network health check from x.x.x..104
        if(! defined($what) && $remote_host_ip !~ /209.10.35.104/){
            print "$short_name@$server_port encountered a client who said nothing! $remote_host_ip , $date<-- \n "
;
            exit(1);
        }
        $first_line = $what;

        if($debug){ print "\n First line is \"", $what, "\" \n"; }
        if ($what !~ /^(GET|HEAD)\s+/i) { #valid requests must start with GET or HEAD
            if($remote_host_ip !~ /209.10.35.104/o){
                print "$short_name -> bad req HTTP/1.1 400, no GET HEAD, $remote_host_ip said:->$what<- \n";
            }
            print $client "HTTP/1.1 400, \"Bad Request\"";
            exit(1);
        }
    }
}

```

```

if ($what == /$mp3_root_dir/i) {
    print "$short_name -> bad req HTTP/1.1 400, mp3_dir in request, $remote_host_ip said:->$what<- \n";
    print $client "HTTP/1.1 400, "Bad Request";
    exit(1);
}

```

#banned ip list

```

if ($remote_host_ip =~ /132.239.12.77/
    || $remote_host_ip =~ /66.122.240.11/
    || $remote_host_ip =~ /63.17.233.236/) {
    #print " $short_name sending HTTP/1.1 400, banned $remote_host_ip \n said:->$what<- \n";
    print $client "HTTP/1.1 400, "Bad Request";
    exit(1);
}

```

#now parse line, extract what we need or dump 'em

```

if ($what =~ /^(([A-Z]+)\s+([\S]+)\s+HTTPV([\d.]+\s*)/){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;
} else {
    print " $short_name: sending bad client HTTP/1.1 400, Bad Request message\n";
    print " $short_name: bad req was: $what\n";
    print $client "HTTP/1.1 400, "Bad Request";
    exit(1);
}

```

print "method is > \$what_method< song is >\$what_song< version is >\$what_version< \n";

```

if ($what_method !~ /(GET|HEAD)/i) {
    exit(1);
}

```

#now loop to get other info lines

```

while( ($request = <$client>) ne ("r\n" || "r\nr\n") ){
    last if(! defined($request));
    $request =~ s/[r\n]/g;
    if($debug) {
        print "<$level: $counter>The client said: " . $request . "<\n";
    }
    #----- WEB BROWSER INFO ----- #
    if($request =~ /Mozilla/io) {
        $ok = 0; $mozzy = 1; $request =~ /.+:\s*(.+)/; $ua = $1;
    }
    if($request =~ /Opera/io) {
        $ok = 0; $mozzy = 1; $request =~ /.+:\s*(.+)/; $ua = $1;
    }
    if($request =~ /Omniweb/io) {
        $ok = 0; $mozzy = 1; $request =~ /.+:\s*(.+)/; $ua = $1;
    }
    if($request =~ /iCab/io) {
        $ok = 0; $mozzy = 1; $request =~ /.+:\s*(.+)/; $ua = $1;
    }
    #----- MP3 RIPPER INFO ----- #
    if($request =~ /StreamRipper/io) {

```

```

        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- No get for Wget ----- #
    if($request =~ /Wget/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- FlashGet fizzles ----- #
    if($request =~ /FlashGet/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- GetRight is Wrong ----- #
    if($request =~ /GetRight/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- Step on Spiders ----- #
    if($request =~ /asterias/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- Web access = No access ----- #
    if($request =~ /Web access/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }

    #----- Monica gets the bone ----- #
    if($request =~ /monica/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- man what the hell is this? ;- ) ----- #
    if($request =~ /Media Jukebox WinInet Reader/io){
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- burn Nero Player ----- #
    if($request =~ /NeroMediaPlayer/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- 1SMUSIC is deaf ----- #
    if($request =~ /1SMUSIC/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }

    #----- I know who Anonymizer is ----- #
    if($request =~ /Anonymizer/io) {
        $ok = 0; $ripper = 1; $request =~ /\.*:s*(.+)$/; $ua = $1;
    }
    #----- MP3 CLIENT INFO ----- #
    if($request =~ /Windows-Media-Player/io){
        $ok = 0; $nsplayer = 1; $request =~ /\.+:s*(.+)$/; $ua = $1;
    }
    if($request =~ /NSPlayer/io) {
        $ok = 0; $nsplayer = 1; $request =~ /\.+:s*(.+)$/; $ua = $1;
    }
    if($request =~ /RMA/io) {
        $ok = 0; $real = 1; $request =~ /\.+:s*(.+)$/; $ua = $1;
    }
    if($request =~ /Winamp/io) { $ok = 1; $request =~ /\.+:s*(.+)s*$/; $ua = $1; }
    if($request =~ /QuickTime/io) { $ok = 1; $request =~ /\.+:s*(.+)$/; $ua = $1; }

```

```

if($request =~ /Xaudio/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}
if($request =~ /AppleApp/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}
if($request =~ /iTunes/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}
if($request =~ /Sonique/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}
if($request =~ /xmms/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}
if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)\S/; Sua = $1;}

#### hold for error reportin ===#
if($request =~ /User Agent/io) { $u_a_line = $request; }

last if ($counter > 20 ); #we don't want some client babbling on and on
$counter++;

}#end while

if($debug){
    print "User Agent = $ua\n";
    print "request = $what_song\n";
}

# ##### Rate controls to prevent excessive access to content

$remote_host_ip =~ /\.+\.+/;
# my $last_octet_mod = $1;
# $last_octet_mod = $last_octet_mod % 1; #mod by number of tables we want
# my $time_offset = 1;

my $sql0 = "Select count(ip_address) as loser_count from loser ";
$sql0 = "where ip_address = \"$remote_host_ip\" ";
$sql0 = "AND user_agent = \"$ua\"";
my $sth0 = $dbh->prepare($sql0);
$sth0->execute();
my $ref0 = $sth0->fetchrow_hashref();
my $loser_count = $ref0->{'loser_count'};
$sth0->finish();

##
if($loser_count <= 0 ){

my $sql = "SELECT COUNT(request_time) as number_songs_reqs ";
#$sql = "MINUTE(MIN(request_time)) - MINUTE(now()) as duration ";
$sql = "FROM rate_control_0 ";
$sql = "WHERE ";
$sql = "ip_address = \"$remote_host_ip\" ";
$sql = "AND ";
$sql = "user_agent = \"$ua\" ";
$sql = "AND ";
$sql = "request_time > now() - INTERVAL 60 SECOND";

print "\n$sql\n" if $debug;

my $sth = $dbh->prepare($sql);
$sth->execute();

my $ref = $sth->fetchrow_hashref();

```

```

my $song_count = $ref->{'number_songs_reqs'};
#my $duration = ($ref->{'duration'});
$sth->finish();

print "\n Song count is $song_count \n" if $debug;

if($song_count >= $limit){
    $speeder = 1;
    $ok = 0;
    print "\n Loser exceeded limit\n" if $debug ;

    my $sql1 = "insert into loser (ip_address,user_agent,loser_time) values (\'$remote_host_ip\','$ua',now()
)";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if song_count

} else{
    print "\n loser is in loser table\n" if $debug;
    $speeder = 1 ;
    $ok = 0;
}

if(!$speeder){
    my $sql1 = "INSERT INTO rate_control_0 ";
    $sql1 .= "(ip_address,user_agent,request_time) ";
    $sql1 .= "VALUES (\'$remote_host_ip\','$ua',now())";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if

print $client "HTTP/$what_version 302 Found \r\n";
print " I told him HTTP/$what_version 302 Found \n" if $debug;
print $client "Date: $date \r\n";
print $client "Server: Apache/1.3.20 \r\n"; #lie to 'em to keep 'em guessing.
print $client "Keep-Alive: timeout=10, max=20\r\n";

#----- Good guys ----- #
if(($nsplayer || $real) && ! $speeder){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    print "found nsplayer or real player\n" if $debug;
}
if($ok && ! $speeder){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($mozzzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    # print "mozzy type web server being redirected to index page\n";
}

```

```

    }
    if($ripper){
        print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
        # print "mozzy type web server being redirected to index page\n";
    }
    if( $speeder ){
        print "Sshort_name: Speeder @ $remote_host_ip sending bad client HTTP/1.1 400, Bad Request
message\n";
        print "Sshort_name: Speeder said $what\n";
        print $client "HTTP/1.1 400, \"Bad Request\"";
    }

    $ok = 1;
    $nsplayer = 0;
    $mozzy = 0;
    $real = 0;

    print $client "\r\n\r\n";
    # print "\ndone\n";
    # print "*****\n\n";

#    $dbh->disconnect();
    close $client;

    $what_song =~ /(.(+).mp3)/io;
    $what_list = $1;
    $song_name = $2;

#trash filter *****
    if(! defined($what_list)){
        print "Sshort_name: no list\n"; $what_list = "none\n";
    }
    if(! defined($song_name)){
        print "Sshort_name: no song_name $remote_host_ip\n"; $song_name = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "Sshort_name: no hostname\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($remote_host_ip)){
        print "Sshort_name: no host_ip\n"; $remote_host_ip = "none";
        $bad = 1;
    }
    if(! defined($remote_hostname)){
        print "Sshort_name: no hostname $remote_host_ip\n"; $remote_hostname = "none";
        $bad = 1;
    }
    if(! defined($ua)){
        print "Sshort_name: no ua\n"; $ua = "none";
        $bad = 1;
    }
}

#match this: APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}\"i\" \"%{User-agent}\"i\")
#change date from: Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

```

```

Sdate =~ /.+\s(.+)\s(.+)\s(.+)\s(.+)/io;
my $month_name = "$1";
my $day = "$2";
my $time = "$3";
my $year = "$4";

if ($day < 10) { $day = "0$day"; } #hack to add leading zero

$month_name =~ s/ //g;
#my $apache_date = "[ $day/$1/$4:$3 -0500]";
my $apache_date = "[ $day/$month_name/$year:$time -0500]";
$whole_request =~ s/r//;
$whole_request =~ s/\n//;

open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
$sa_str = "$remote_host_ip - - $apache_date \"$whole_request\" 302 2 \"$program_name\" \"$ua\"";
print APACHE_LOG "$sa_str\n";
close APACHE_LOG;

if ($bad || $ua eq "unknown" ) {
    $first_line =~ s/[\r\n]//g;
    open (ERROR_LOG, ">>$error_log");
    my $se_str = "$remote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
    print ERROR_LOG "$se_str\n";
    close ERROR_LOG;
} #end if error

close STDOUT;
close STDERR;
close STDIN;

exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
$dbh->disconnect();
close ($server);
exit(0);

```

Operation of Chooser Script

Purpose:

To provide guests a user selectable variety of the "Music at The MoMI" Encoded music files, by allowing the guest to select individual songs from all of the available lists present at this location for their listening pleasure.

Background:

Typically, a site that hosts music files provides a vast quantity of links that are generically organized by musical style or genre. These are, in turn, links to the files that will be played. Selections may be saved on some sites, but the sites do not have any mechanism for allowing a user based selection in the form of a unique list selected from the overall musical content at that location, but rather the selections are usually single-play choices, and when a saved list approach is used, the user must traverse the entirety of the site to select the musical choices.

Prior Implementations:

MyMP3.com is the closest implementation that could mimic the functionality of the MoMI site. A user may select a low-fi (56K encoded music file) version for saving as a cookie based list for future use. The list is perpetual, meaning that the list is the list until the user edits out the songs that they no longer want on the list. There is only one list saved, and the list must be played from that location while on their site.

Preferred Embodiment:

The musical play lists (lists of songs available) of all varieties are loaded from the MoMI server and passed to a Common Gateway Interface (CGI) program that is written in Perl Script, where the lists are concatenated into a single dimensioned array.

In contrast to prior implementations, the MoMI site provides (via the choose.cgi program) the ability to select songs at whimsy from the available content, and save them for future re-use. Alternately, they can be executed immediately and not saved. A guest of the MoMI site could save hundreds of these customized playlists to their desktop, and play them at will. The advantage here is that the user has a reason to return again and again, based on the depth of the musical selection, to select more variations in listening.

The actual code is provided as appendix 2-II.

Appendix 2-II

```

#!/usr/bin/perl -w
#-----
#           User choice Program
# This program reads in a mp3 play list source file, displays the file names,
# on a web page to allow for selection, and creates a new playlist file to download.
# Programmer: Ted Fitzgerald, ICS CREATIVE
# Initial Creation Date: July 26, 2001
# Created for: "themomi.org"
# version 1.0 proof of concept / prototype / extremely easy to understand code
# version 2.0 port to unix, localize to utilize site dir structure
# version 2.1 rewrite to include customer design changes,
#       display cover.jpg's and m3u file found in each sub dir mp3 directory.
# version 2.2 add fix for malformed .m3u files
# version 3.0 change display format to 2 column mode
# version 4.0 add select all in category button and supporting logic.
# version 4.1 add image for select button and add all supporting logic.
#       We now need to look for param(Submit.x) for a submitted pagea
# version 5.0 Complete rewrite to use database
#-----
$VERSION = "5.0";

$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# ---- declare some variables used in the code program ----- #
# ---- Don't touch these --- #

my $dir_cnt = 0; #counter
my $qry1 = "";
my $qry2 = "";
my $sql = "";

# -----YOU CAN/SHOULD LOCALIZE THESE VARIABLES -----
#

my $rootdir = "mp3";
my $image_path = "http://www.themomi.org/museum/images";
my $image_url = "www.themomi.org/$rootdir";
my $song_root = "../htdocs/";
my $earthc = "themomi.earthc.net";

# ----- Start Code ----- #

$|=1; #flush
print "Content-type: text/html\n\n"; # send basic header
use CGI qw(:standard);

```

```

use CGI::Carp('fatalsToBrowser');
use DBI();
use strict;

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "NOBODY", "ASK_FOR_JOE",
                      {'RaiseError' => 1}) or die "can't connect to db";

my @play_list;
my $artist;
my $song_name;

print<<HTML_DONE;
  <html>
  <head>
  <META NAME="Organization" CONTENT="www.themomi.org">
  <META NAME="Author" CONTENT="Ted Fitzgerald">
  <META NAME="Quote" CONTENT="My hovercraft is full of eels!">
  <META NAME="Description" CONTENT="Music selection / creation software">
  <SCRIPT LANGUAGE="JavaScript">
  <!--
    function changeImages() {
      for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
      }
    }
  // -->
  </SCRIPT>
  <style><!--a{text-decoration:none}!--></style>
  <style><!--a:hover{color:#3399cc; }--></style>

  </head>
  <body bgcolor="black">
  <font size=2 color=white face = arial,san serif, helvetica >
  <form method="post">

HTML_DONE

#----- Main page display starts here (selection portion ----- #

# if first time page is displayed submit button hasn't been clicked and submit.blah is
undefined.
# we use this parameter to show initial selection or results portion of page

if(! defined(param("Submit.x"))){ #if first time page is displayed submit is undef

```

```

print<<HTML_DONE;
    <center>
    <a id="button">&nbsp;</a>
    <a name="button">&nbsp;</a>
    <input type="image" value="Submit" src="$image_path/oy.gif" border="0"
alt="Create Playlist" name="Submit">
    <!-- <A HREF="http://www.themomi.org/perl/choose_hi.cgi"
        ONMOUSEOVER="changeImages('oy',
'http://www.themomi.org/museum/images/oy.gif'); return true;"
        ONMOUSEOUT="changeImages('oy',
'http://www.themomi.org/museum/images/oy_over.gif'); return true;">
    <IMG NAME="oy" SRC="http://www.themomi.org/museum/images/oy.gif"
BORDER=0></A> -->
    <br><br>
    <input type=Reset value="Reset" name=Reset>
    </center>
HTML_DONE

# ----- MAIN CODE STARTS HERE ----- #

my $sql = "select distinct playlist_name from m3u";
    $sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
    $sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);

    my $even = $dir_cnt % 2; # flip flop the output to make 2 columns odd = left even =
right

    #make left right columns
    if(($even)) {
        print "<td align=center valign=top width=50%>";
    } else {
        print "<table align=center border=0 width=90% bordercolor=black name=outer>";
#start outer table
        print "<tr><td align=center valign=top width=50%>";
    }# end if

    print "<br><br><img src=http://$searchc/$image_url/$playlist_name/cover.jpg
align=center><br><br>";

    print "<table border=0 align=center width=90% bordercolor=black
name=internal_table>";

```

```

print "<tr><td><input type=checkbox name=d$dir_cnt></td>";
print "<td><font size=2 color=white face = arial,san serif, helvetica >";
print " --- Select All in Category ---</font>";
print "</td></tr>";

my $sql1 = "select pl_index,artist,song_name from m3u where playlist_name like
\"$playlist_name\"";
# print "$sql1<br>";
my $sth1 = $dbh->prepare($sql1);
$sth1->execute();

while (my $ref1 = $sth1->fetchrow_hashref()) {
    my $id = $ref1->{'pl_index'}; $artist = $ref1->{'artist'}; $song_name = $ref1-
>{'song_name'};
    $artist =~ s/_/_/g;
    $song_name =~ s/_/_/g;
    print "<tr><td><input type=checkbox name=\"$id\"></td>\n";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$song_name<br></font></td></tr>\n";
} #end while ref1

print "</table>"; # end inner table

if($seven) {
    print "</td></tr>";
    print "</table>"; #end outer table
    print "<center><br><font size=2 color=white face = arial,san serif, helvetica >";
    print "<a href=\"\"#button\">Return to top</a></font></center>";
} else {
    print "</td>";
} #end ending left right column if

$dir_cnt++; # increment counter used for left/right flip flop

} #end while ref -> display of play lists loop
#-----
print "<br><br>";

print "<input type=hidden name=size value=$dir_cnt>";

# $dbh->disconnect();

print " </form>";

```

```

} #end if (Submit)
#-- done displaying the selection list --

```

```

#-----#
#----- DISPLAY RESULTS OF SELECTIONS -----#
#
#-----#

```

```

if(defined(param("Submit.x")) || defined(param("Submit.x"))){
  my @play_list;
  my @list_copy;
  my @p_list;
  my $pl;
  my $first_pl;
  my $qry1;
  my $qry2;
  # $num_of_dirs = param("size");

```

```

#

```

```

=====
#my $sql = "select distinct playlist_name from m3u order by playlist_name asc";
my $sql = "select distinct playlist_name from m3u";
  $sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
  my $playlist_name = $ref->{'playlist_name'};
  push(@play_list,$playlist_name);
} #end while "get all playlists" query
#
=====

```

```

#-----#
# parse param list for "all in playlist" request = "d+some num" & individual songs "num"
#-----#
my @song_list;
#my $pl_limit = 5;
#my $song_limit = 50;
#my $pl_count = 1;

foreach (param()) {
  push(@song_list,$_) if /^d+$/;    #just a song number
  # print "<br>$_" if /^d+$/;

```

```

if (/^d\d+$/){          #all in playlist d+number
    s/^d//;             #remove letter "d"

    push(@p_list,$play_list[$_]); #store playlist number
    push(@list_copy,$play_list[$_]);

    } # end if "d+numb all in playlist "
} # end foreach "item in param() list"
# -----
#      Build "All in playlist" query string
# -----

if(@p_list){

    my $first_pl = shift @p_list;
    my @list_copy = $first_pl;
    $qry1 = "SELECT * FROM m3u WHERE playlist_name IN ( \" $first_pl\" \" \" ;

    foreach my $pl (@p_list){
    #      last if($pl_count = $pl_limit);
        #print "<br>=". @p_list; #print "<br>@p_list";

        $qry1 .= ", \" $pl\" \" ";

        push(@list_copy,$pl);
    #      $pl_count++;
    } #end foreach loop to build items in query string

    $qry1 .= ") order by playlist_name"; #      query ends here

    #print "<br>$qry1";

} #end if anything in p_list array

# -----
#      Build individually selected songs query string
# -----
if(@song_list){ #anyone home in song lists? go for it!

    my @song_copy = @song_list;

    my $first = shift @song_list;
    $qry2 = "SELECT * FROM m3u WHERE pl_index IN ( $first\" ;
    foreach my $blarg (@song_list){
        #print "<br>=". @song_list;
        #print "<br>@song_list";
    }
}

```

```

    $qry2 .= ", $blarg ";
} #
$qry2 .= ")";
# $qry2 .= " limit $song_limit"; #add limit to number of songs
# print "<br>$qry2";
} #end if anything in song_list array

# -----
#   Now create m3u playlist file
#   Execute query statement(s) and write file
# -----

# print "<br>here's your check of array size greater than zero
defined($qry1),defined($qry2)<br>";
if($qry1 || $qry2){
    # ---- Create unique play list name using numbers from date parts ----- #
    my $theDate = localtime;
    $theDate =~ /(\d+):(\d+):(\d+)\b/i;
    my $play_list_name = "Momi_playlist_$2$3.m3u";
    # ----- #

    print "<center><font size=2 color=white face = arial,san serif, helvetica >";
    print "<b>Songs in your play list:</b></font></center>";
    print " <table align=center>";

    open(OUTFILE,">../htdocs/temp/$play_list_name");

    print OUTFILE '#EXTM3U'."\n"; #write file header
    #print '<br>#EXTM3U<br>'; #write file header

    if($qry1){
        my $list = "";
        # now do the playlists
        my $sth1 = $dbh->prepare($qry1);
        $sth1->execute();
        while (my $ref1 = $sth1->fetchrow_hashref()) {
            my $playlist_name = $ref1->{'playlist_name'};
            my $pl_header = $ref1->{'pl_header'};
            my $song_url = $ref1->{'song_url'};
            $song_url =~ s/8080/8081/g;
            print OUTFILE "$pl_header\n$song_url\n";
        } # end while
        #print "<b>my list_copy array = @list_copy</b><br>";
        foreach my $list (@list_copy){
            $list =~ s/_/_/g;

```

```

        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >All Songs
from <b>\'$list\'</b></font></td>
></tr>\n";
    } # end foreach
} #end if $qry1 is defined

if($qry2){
    # now do the selected songs
    my $sth2 = $dbh->prepare($qry2);
    $sth2->execute();
    while (my $ref2 = $sth2->fetchrow_hashref()) {
        my $artist = $ref2->{'artist'};
        my $song_name = $ref2->{'song_name'};
        my $pl_header = $ref2->{'pl_header'};
        my $song_url = $ref2->{'song_url'};
        # $print "$pl_header<br>$song_url<br>" ;
        $artist =~ s/_/_/g;
        $song_name =~ s/_/_/g;
        $song_url =~ s/8080/8081/g;
        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$song_name</font></td></tr>";
        print OUTFILE "$pl_header\n$song_url\n";
    } # end while
} # end if $qry2 is defined

close OUTFILE;

print "</table>";
print "<center><br><br><br>";
print "<a href=\"http://www.themomi.org/temp/$play_list_name\">";
print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
print "</font>";
print "<br><br><br>";

# ----- include text for nubies ----- #

print "<br><br>";
print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the button above to launch
your playlist<br> with your favorite .mp3/.m3u player
</td>

```

```
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>
```

```
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
```

```
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the button above and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
```

```
<tr><td colspan=3 align=center>
MORE
```

```
#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
```

```
print "</table><br><br>";
```

```
}else{
  print "<br><b>Nothing Selected</b><br>";
```

```
  #foreach (param()) {
  #   print "$_<br>";
  #}#end foreach
  #print "The value of Param(\"Submit.x\") is " , param("Submit.x") , "<br>";
  #print "The value of defined Param(\"Submit.x\") is " , defined(param("Submit.x")),
  "<br>";
```

```
  $dbh->disconnect();
```

```
}#end if defined queries
}# end if (Submit.x) "display results of selection "
```

```
print "</body>";
print end_html();
```

```
#bye bye! Thanks for visting!
```

```
exit(0);
```

```
1;
```

Operation of Randomizer Script

Purpose:

To provide guests a varied selection of the "Music at The MoMI" Encoded music files, by randomly selecting a group of files for their listening pleasure from all of the available lists present at this location.

Background:

Typically, a site that hosts music files provides a vast quantity of links that are generically organized by musical style or genre. These are, in turn, links to the files that will be played. Selections may be saved on some sites, but the sites do not have any mechanism for allowing a random selection of the overall musical content at that location.

Prior Implementations:

None

Preferred Embodiment:

The musical play lists (lists of songs available) of all varieties are loaded from the MoMI server and passed to a Common Gateway Interface (CGI) program that is written in Perl Script, where the lists are concatenated into a single dimensioned array. Using a random number generator that incorporates as its product the combination of:

- 1) The number produced by the random function (not a truly random number, but random sequence that follows the "seeded" number. This seeded number is a starting point for the further permutations of random selection, but would have been predictable with a fixed seed value.)
- 2) The Julian date in the server including the hours, minutes, seconds, day, date, and four-digit year.

Using that combination, the following code is executed:

```
$theDate = 'date'; #get system date
$theDate =~ /(d+):(d+):(d+)\b/i;
$play_list_name = "Momi_random_playlist_$2$3.m3u";

$random_seed = "$3$2$3"; #use time parts to make random
srand $random_seed; # seed the random num gen.
```

As can be seen, the time/date then provides a basis for seeding the random number generator that is only repeatable once per millennium. Now that a random number has been selected, the actual process of applying this to the array of song titles begins.

The criteria for selecting a song for inclusion in the random play list is three-fold:

- 1) There cannot be more than 50 songs total in the list;
- 2) The selection must come from no more than 20 play lists
- 3) The selection must not have been previously selected.

So assuming that the first two conditions have not been met, the resulting random number is used as the index into the array and will select the song title that resides in that location in the array. If that title has been previously selected, a new number is requested, and the process continues until all 3 conditions are fulfilled.

The actual code is provided as appendix 2-III.

The resulting list of song titles are then formatted into a usable playlist (or M3U file as it is known), and it is written to the server's hard disk while a page is prepared for presenting this to the user.

The code then formats an HTML page for the viewer to see that contains a link to the M3U file that is on the MoMI server, which they can either execute immediately, or download. A downloaded playlist cannot provide information to the user regarding the real location of the files, as the original playlists do not contain that information. (see document on the redirector script).

Appendix 2-III

```

#!/usr/bin/perl -w
#-----
#
#          RANDOM PLAY LIST PROGRAM
#
# This program reads in a mp3 play list source file in many directories ,
# displays the random choice of file names ,
# and creates a new file to download.
# Programmer: Ted Fitzgerald, ICS Creative
# Date: July 26, 2001
# Created for "themomi.org"
# version 1.0 proof of concept
# version 2.0 port to unix , rewrite to include customer design changes
#      display cover.jpg's and m3u file found in each sub dir mp3 directory.
# version 2.1 added more randomness to the choices
# version 3.0 rewrote the code to use the cgi O.O mode for speed improvements.
# version 4.0 rewrote the code to increase the randomness and limit the number of
#      selections that are possible. Added tons of comments.
# version 5.0 rewrote app to use the mysql database, March 17 2002
#
#-----
my $version;
$version = 5.0;
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# declare variables

my ($i,$j); #just counters
my ($play_list_name); #variable to create random file name for download
my ($show_many_songs); # = total number of songs in all the dirs.
my (@thename); #array to contain the name of the randomly selected song
my (@thepath); #array to contain the URL path of the randomly selected song
my ($showname); #used to find the portion of name we want to display.
my ($song_url);
my (@theheader);
my (@artist);
my ($artist);
my ($theDate);
# ----- user ajustable paramerters -----
my $max_num_songs = 34; # set limit of how many songs to display
my $maxdirs = 20; #set limit on number of dirs to look in
my $mp3dir = "../htdocs/mp3"; #path from cgi script to mp3 directory
my $center_col_size = ($max_num_songs / 2 + 1);

$|=1; #flush the buffer
use strict;

```

```

use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
    "PIZZA_MAN", "HE_DELIVERS",
    {'RaiseError' => 1}) or die "can't connect to db";

my $q = new CGI;

print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>Random Play List Creation</title>";
print "<style><!--a{text-decoration:none}!--></style>";
print "<style><!--a:hover{color:#3399cc; }--></style>";

print <<HTML_DONE;
<SCRIPT LANGUAGE="JavaScript">
<!--

function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}

// -->
</SCRIPT>
<head>
<body bgcolor="black" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="2" text="white" link="#ffcc66" vlink="#ffcc
66" alink="#ffcc66">

<font size=2 color=white face = arial,san serif, helvetica >
HTML_DONE

$theDate = localtime; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "Momi_random_playlist_$3$2$3.m3u";
#-----#
# do all file io before displaying the names in list

```

```
my $sql = "select pl_header,song_name,song_url,artist from m3u order by rand() limit
$max_num_songs";
```

```
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $pl_header = $ref->{'pl_header'};
    my $song_name = $ref->{'song_name'};
    my $song_url = $ref->{'song_url'};
    my $artist = $ref->{'artist'};
    push (@thename,$song_name);
    push (@thepath,$song_url);
    push (@theheader,$pl_header);
    push (@artist,$artist);
}
```

```
open(OUTFILE,">./htdocs/temp/$play_list_name");
#write out file header
print OUTFILE "#EXTM3U\n";
for($i=0;$i<$max_num_songs;$i++){
```

```
    print OUTFILE "$theheader[$i]\n";
    $song_url = "$thepath[$i]";
    $song_url =~ s/8080/8081/io;
    print OUTFILE "$song_url\n";
```

```
}#end for
close OUTFILE;
```

```
#-----#
#now show 'em what they have in the list
```

```
print<<MORE;
```

```
<center>
```

```
MORE
```

```
print "<br>";
```

```
print "</center>";
```

```
print "<table align=\"center\" border=0 width = \"95%\">";
```

```
print "<tr width=45%><td><td rowspan=$center_col_size align=center valign=top>";
```

```
print <<HERE;
```

```

<A HREF="http://www.themomi.org/temp/$play_list_name"
  ONMOUSEOVER="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert_over.gif');
return true;"
  ONMOUSEOUT="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert.gif'); return
true;">
  <IMG NAME="create_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/create_now_vert.gif"
BORDER=0></A>
HERE

print "</td>";
print "<td width=45%></td></tr>";
for($i=0;$i<$max_num_songs;$i++){
  my $flip_flop = ($i % 2 );

  $showname = $thename[$i];
  #$showname =~ /EXTINF:\d+,(.+)/i;
  $showname =~ s/_/ /g;
  $artist = $artist[$i];
  $artist =~ s/_/ /g;

  if($flip_flop){
    print "<td width = 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showname
</font></td></tr>";
  }else{
    print "<tr><td width= 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showna
me </font></td>";
  }
}

print "</table><br>";

#-----#

# print "<center>";
# print "<br><a href=\"http://www.themomi.org/temp/$play_list_name\">";
# print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
# print "</center>";
print "<table align=center border=0 width=90%>";
print<<MORE;

```

```

<tr><td width=40% align=center>&nbsp;</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">
MORE

print "<A HREF=\"http://www.themomi.org/perl/random_hi.cgi\"";

print<<MORE:
  ONMOUSEOVER="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm_over.gif');
  return true;"
  ONMOUSEOUT="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm.gif');
  return true;">
    <IMG NAME="rerandom_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/re-randomize_sm.gif"
BORDER=0></A>
</font></td>
<td>&nbsp;</td>
</tr>
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the center button above to launch
your <br>playlist with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the center button and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
<tr><td colspan=3 align=center>
MORE

#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
print "</table><br><br>";
print "<center>";

print $q->end_html;

1;

```

Section 3: METHOD AND SYSTEM FOR PROVIDING LOCATION-OBSCURED
MEDIA DELIVERY

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

NOTATION AND NOMENCLATURE

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital system memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions

leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that throughout discussions of the present invention, discussions utilizing terms such as "providing" or "determining" or "activating" or "controlling" or "transmitting" or "receiving" or "recognizing" or "generating" or "utilizing" or "storing" or the like, refer to the action and processes of a computing system, or similar electronic computing device, that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system memories or registers or other such information storage, transmission, or display devices.

EXEMPLARY HARDWARE IN ACCORDANCE WITH THE PRESENT INVENTION

Figure 3-1 is a block diagram of an embodiment of an exemplary computer system 3-100 that may be used in accordance with the present invention. It is understood that system 3-100 is not strictly limited to be a computer system. As such, system 3-100 of the present embodiment is well suited to be any type of computing device (e.g., server computer, desktop computer, laptop computer, portable computing device, etc.). Within the discussions of the present invention, certain processes and steps are discussed that are realized, in one embodiment, as a series of instructions (e.g., software program) that reside within computer readable memory units of computer system 3-100 and executed by a processor(s) of system 3-100. When executed, the instructions cause computer 3-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 3-100 of Figure 3-1 comprises an address/data bus 3-110 for communicating information, one or more central processors 3-102 coupled with bus 3-110 for processing information and instructions. Central processor unit(s) 3-102 may be a microprocessor or any other type of processor. The computer 3-100 also includes data storage features such as a computer usable volatile memory unit 3-104, e.g., random access memory (RAM), static RAM, dynamic RAM, etc., coupled with bus 3-110 for storing information and instructions for central processor(s) 3-102, a computer usable non-volatile memory unit 3-106, e.g., read only memory (ROM), programmable ROM, flash memory, erasable programmable read only memory (EPROM), electrically

erasable programmable read only memory (EEPROM), etc., coupled with bus 3-110 for storing static information and instructions for processor(s) 3-102.

System 3-100 also includes one or more signal generating and receiving devices 3-108 coupled with bus 3-110 for enabling system 3-100 to interface with other electronic devices. The communication interface(s) 3-108 of the present embodiment may include wired and/or wireless communication technology. For example, in one embodiment of the present invention, the communication interface 3-108 is a serial communication port, but could also alternatively be any of a number of well known communication standards and protocols, e.g., a Universal Serial Bus (USB), an Ethernet adapter, a FireWire (IEEE 1394) interface, a parallel port, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In another embodiment a digital subscriber line (DSL) connection may be employed. In such a case the communication interface(s) 3-108 may include a DSL modem. In any case, the communication interface(s) 3-108 may provide a communication interface to the Internet.

Optionally, computer system 3-100 can include an alphanumeric input device 3-114 including alphanumeric and function keys coupled to the bus 3-110 for communicating information and command selections to the central processor(s) 3-102. The computer 3-100 can also include an optional cursor control or cursor directing device 3-116 coupled to the bus 3-110 for communicating user input information and command selections to the central processor(s) 3-102. The cursor directing device 3-116 can be implemented using

a number of well known devices such as a mouse, a track ball, a track pad, an optical tracking device, a touch screen, etc. Alternatively, it is appreciated that a cursor can be directed and/or activated via input from the alphanumeric input device 3-114 using special keys and key sequence commands. The present embodiment is also well suited to directing a cursor by other means such as, for example, voice commands.

The system 3-100 of Figure 3-1 can also include a computer usable mass data storage device 3-118 such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) coupled with bus 3-110 for storing information and instructions. An optional display device 3-112 is coupled to bus 3-110 of system 3-100 for displaying video and/or graphics. It should be appreciated that optional display device 3-112 may be a cathode ray tube (CRT), flat panel liquid crystal display (LCD), field emission display (FED), plasma display or any other display device suitable for displaying video and/or graphic images and alphanumeric characters recognizable to a user.

EXEMPLARY NETWORK IN ACCORDANCE WITH THE PRESENT INVENTION

Figure 3-2 is a block diagram of an exemplary network 3-200 that may be used in accordance with an embodiment of the present invention. For example, network 3-200 of the present embodiment enables one or more client computer systems (e.g., 3-204, 3-206 and/or 3-208) to receive high fidelity media content "on-demand" from a media content server 3-212 via the Internet 3-202 while

restricting unauthorized users from directly retrieving media content from its source database stored by content server 3-212.

Network 3-200 of Figure 3-2 includes web application server 3-210 and content server 3-212 which are communicatively coupled to the Internet 3-202. Additionally, web application server 3-210 and content server 3-212 may be communicatively coupled together within out utilizing Internet 3-202. Furthermore, network 3-200 includes client computers 3-204, 3-206 and 3-208 which are each communicatively coupled to the Internet 3-202. In this manner, web server 3-210, content server 3-212, and client computers 3-204 to 3-208 are able to communicate. It should be appreciated that the devices of network 3-200 of the present embodiment are well suited to be coupled in a wide variety of implementations. For example, content server 3-212, web server 3-210, and client computers 3-204 to 3-208 of network 3-200 may be coupled via wired communication technology (e.g., coaxial cable, copper wire, fiber optics, and the like) and/or wireless communication technology.

Within network 3-200, it is understood that web server 3-210, content server 3-212 and client computers 3-204 to 3-208 may be implemented in a variety ways in accordance with the present embodiment. For example, web server 3-210, content server 3-212 and client computers 3-204 to 3-208 may be implemented in a manner similar to computer system 3-100 of Figure 3-1. However, these devices of network 3-200 are not strictly limited to such implementation. Additionally, web server 3-210 and content server 3-212 of the present embodiment perform a variety of functionality within network 3-200. It is

understood that web server 3-210 and content server 3-212 may actually reside on a single physical computing device (e.g., computer 3-100 of Figure 3-1). However, web server 3-210 and content server 3-212 of the present embodiment may each be implemented as one or more physical computing devices.

It is appreciated that network 3-200 of Figure 3-2 is able to operate with and provide delivery of any type of media content (e.g., audio, video, multimedia, graphics, information, data, software programs, and/or the like) in any type of format. For example, the content server 3-212 may provide audio clips, video clips, and the like to client computers 3-204 to 3-208 via the Internet 3-202.

EXEMPLARY OPERATIONS IN ACCORDANCE WITH THE PRESENT INVENTION

Figure 3-3 is a block diagram of an exemplary system 3-300 for implementing access keys in accordance with an embodiment of the present invention. Specifically, system 3-300 illustrates a client computer system (e.g., 3-204) utilizing access keys in order to receive high fidelity media content "on-demand" from a media content server (e.g., 3-212) while unauthorized users or media applications are restricted from directly retrieving and/or copying media content from its media source database (e.g., 3-304).

The client computer device 3-204 may communicatively couple with a web/application server (e.g., 3-210) in order to request a media play list. In response, the web server 3-210 may determine whether the user of client 3-204 is authorized to receive the media play list associated with the request. The web

server 3-210 may utilize a user database 3-302 in order to determine if a user is authorized to receive a media play list. If not, web server 3-210 does not provide client computer 3-204 the requested media play list. However, if client 3-204 is authorized, web server 3-210 transmits the media play list to the client device 3-204 (e.g., via Internet 3-202) along with an appended user key or user identification (ID). It is understood that user database 3-302 may also include data for one or more media play lists that may be utilized to provide a media play list to client computer 3-204. Subsequently, the user of client device 3-204 may utilize the received media play list in combination with a media player application operating on client device 3-204 in order to transmit a delivery request (that automatically includes the user key or user ID) for one or more desirable pieces of media content to web server 3-210.

Upon reception of the media content delivery request, the web server 3-210 of Figure 3-3 checks the validity of the requesting media application along with the attached user key. The web server 3-210 may utilize user database 3-302 in order to check the validity of the user key (or user ID) and the requesting media application. If either or both are invalid, web server 3-210 redirects the unauthorized client computer 3-204 so that it may be handled in an appropriate manner in order to prevent abuse of the system. However, if both the requesting media application and user ID are valid, web server 3-210 issues to client device 3-204 a redirect command to the current address location of the desired media content along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe).

In response to reception of the redirect command, the media player application operating on client computer device 3-204 automatically issues a new request along with the time sensitive access key to content server 3-212 for delivery of the one or more desired pieces of media content. The content server 3-212 determines the validity of the time sensitive access key. If invalid, content server 3-212 redirects the unauthorized client device 3-204 so that it may be handled in an appropriate manner in order to prevent access to the media content. However, if the time sensitive access key is valid, content server 3-212 retrieves the desired media content from a media content database 3-304 and then transmits it to client device 3-204 for use by its media player application. It is understood that the delivered media content may be stored by client computer 3-204 using hidden directories thereby preventing future unauthorized distribution of it.

Figures 3-4A and 3-4B are a flowchart 3-400 of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device (e.g., 3-204, 3-206 or 3-208). Flowchart 3-400 includes processes of the present invention which, in one embodiment, are carried out by a processor(s) and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions may reside, for example, in data storage features such as computer usable volatile memory 3-104, computer usable non-volatile memory 3-106 and/or computer usable mass data storage device 3-118. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although

specific steps are disclosed in flowchart 3-400, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 3-4A and 3-4B. Within the present embodiment, it should be appreciated that the steps of flowchart 3-400 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment enables delivery of "on-demand" high fidelity media content to computer devices via one or more communication networks (e.g., the Internet) while restricting unauthorized users from directly retrieving media content from a source database. Once the computer device receives the media, it may be stored using hidden directories so that it may not be easily shared with others. Within the present embodiment, there are different functionality that are implemented in order to protect and monitor the media content source. For example, the actual address location of the media database is hidden from content recipients while its address directory is periodically change making past addresses obsolete. Furthermore, an access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media content requests. By implementing these and other functionality, the present embodiment restricts access to and redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

It should be appreciated that flowchart 3-400 is described in conjunction with Figure 3-2 in order to more fully describe the operation of the present

embodiment. At step 3-402 of Figure 3-4A, a client computer (e.g., 3-204) communicatively couples to a web server (e.g., 3-210) via one or more communication networks (e.g., Internet 3-202) and then proceeds to log in. The initial log in process of step 3-402 may be implemented in a wide variety of ways in accordance with the present embodiment. For example, the initial log in process may involve a registration process provided by web server 3-210 wherein different identifying information may be provided by a user of client computer 3-204 such as, but not limited to, his or her name, address, phone number and credit card information. The present embodiment may then cause web server 3-210 to verify the truthfulness of the provided registration information. Furthermore, the registration process may include the user of client device 3-204 establishing with web server 3-210 a user selected identity and password.

However, if the user of client computer 3-204 has already participated in the registration process, the log-in process of step 3-402 may include web server 3-210 requesting that the user of client device 3-204 provide his or her previously established user identity and password. As such, the user of computer 3-204 causes it to transmit his or her user identity and password to web server 3-210. Upon receiving them, web server 3-210 determines their validity. If they are invalid, web server 3-210 refuses access to client computer 3-204 wherein flowchart 3-400 may be discontinued. However, if the provided user identity and password are valid, the present embodiment proceeds to step 3-404.

At step 3-404, once the registration process is completed, the present embodiment causes web server 3-210 to generate a unique user identification (ID)

or user key associated the user of client device 3-204 that participated in the initial log-in process of step 3-402. The unique user ID (or user key) is then stored by web server 3-210 in a manner that it is associated with that registered user. In this manner, the present embodiment is able to uniquely identify each authorized user of web server 3-210. It is appreciated that if a unique user ID has already been generated for a user that logs in with web server 3-210 at step 3-402, the present embodiment may skip step 3-404 and proceed to step 3-406.

In step 3-406 of Figure 3-4A, the user of client computer 3-204 causes it to transmit to web server 3-210 (e.g., via Internet 3-202) a request for a play list of available media content. It is understood that the requested media play list may include all or a portion of the media content available from content server 3-212.

At step 3-408, in response to web server 3-210 receiving the play list request, the present embodiment causes web server 3-210 to transmit to client computer 3-204 a media content play list together with the unique user ID associated with the logged-in client. The user ID may be attached to the media content play list in a manner such that the user of computer 3-204 is unaware of it. It is appreciated that the media content of flowchart 3-400 may include, but is not limited to, high fidelity music, audio, video, graphics, multimedia, and the like. The media content play list of step 3-408 may be implemented in diverse ways. For example, the present embodiment may generate a media content play list by taking all of the media content titles available from the content server 3-212 and combining them into a single list. Alternatively, all of the media content titles (or different lists of titles) may be loaded from content server 3-212 and passed to a

Common Gateway Interface (CGI) program operating on web server 3-210 that may be written in Perl (Practical Extraction and Report Language) Script, where the media titles (or different lists of titles) are concatenated into a single dimensioned array that may be provided to client device 3-204.

In step 3-410 of Figure 3-4A, the user of client computer 3-204 may utilize the received media content play list in conjunction with a media player application in order to cause client device 3-204 to transmit a request to web server 3-210 for delivery of particular media content, wherein the user ID automatically accompanies the request. The media delivery request of the present embodiment may be implemented in a wide variety of ways. For example, the media content play list provided to client device 3-204 by web server 3-210 may enable the user to create one or more customized media play lists by selecting desirable media content titles. It is understood that a customized media play list may establish the media content that eventually will be delivered to client computer 3-204 along with the sequential order in which it will be delivered. Furthermore, the user of client device 3-204 may create as many or as few customized play lists as desired and then store them with computer 3-204 (e.g., to its desktop) and/or within web server 3-210. It is noted that a customized media play list does not actually contain media content, but instead it includes one or more identifiers of specific pieces or portions of media content such as, but not limited to, a song, an audio clip, a video clip, a picture, a graphic picture, a multimedia clip, and the like.

The following Perl Script entitled "User Choice Program" is an exemplary implementation for providing a media play list to enable a user of a computer (e.g.,

3-204) to generate a request for media delivery by creating one or more customized play lists.

```
#!/usr/bin/perl -w
#-----
#           User Choice Program
# This program reads in a mp3 play list source file, displays the file names,
# on a web page to allow for selection, and creates a new playlist file to download.
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
#-----
$VERSION = "5.0";

$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# ---- declare some variables used in the code program ----- #
# ---- Don't touch these --- #

my $dir_cnt = 0; #counter
my $qry1 = "";
my $qry2 = "";
my $sql = "";

# -----YOU CAN/SHOULD LOCALIZE THESE VARIABLES ----- #

my $rootdir = "mp3";
my $image_path = "http://www.themomi.org/museum/images";
my $image_url = "www.themomi.org/$rootdir";
my $song_root = "../htdocs/";
my $earthc = "themomi.earthc.net";

# ----- Start Code ----- #

$|=1; #flush
print "Content-type: text/html\n\n"; # send basic header
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();
use strict;

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
    "NOBODY", "ASK_FOR_JOE",
    { 'RaiseError' => 1 }) or die "can't connect to db";
```

```
my @play_list;
my $artist;
my $song_name;
```

```
print<<HTML_DONE;
  <html>
  <head>
  <META NAME="Organization" CONTENT="www.themomi.org">
  <META NAME="Author" CONTENT="Ted Fitzgerald">
  <META NAME="Quote" CONTENT="My hovercraft is full of eels!">
  <META NAME="Description" CONTENT="Music selection / creation software">
  <SCRIPT LANGUAGE="JavaScript">
  <!--
    function changeImages() {
      for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
      }
    }
  // -->
  </SCRIPT>
  <style><!--a{ text-decoration:none }!--></style>
  <style><!--a: hover{ color:#3399cc; }--></style>

  </head>
  <body bgcolor="black">
  <font size=2 color=white face = arial,san serif, helvetica >
  <form method="post">
```

HTML_DONE

```
#----- Main page display starts here (selection portion ----- #

# if first time page is displayed submit button hasn't been clicked and submit.blah is
undefined.
# we use this parameter to show initial selection or results portion of page
```

```
if(! defined(param("Submit.x"))){ #if first time page is displayed submit is undef
```

```
print<<HTML_DONE;
  <center>
  <a id="button">&nbsp;</a>
  <a name="button">&nbsp;</a>
  <input type="image" value="Submit" src="$image_path/oy.gif" border="0" alt="Create
Playlist" name="Submit">
  <!-- <A HREF="http://www.themomi.org/perl/choose_hi.cgi"
    ONMOUSEOVER="changeImages('oy',
'http://www.themomi.org/museum/images/oy.gif'); return true;"
    ONMOUSEOUT="changeImages('oy',
'http://www.themomi.org/museum/images/oy_over.gif'); return true;">
```

```

    <IMG NAME="oy" SRC="http://www.themomi.org/museum/images/oy.gif"
    BORDER=0></A> -->
    <br><br>
    <input type=Reset value="Reset" name=Reset>
    </center>
HTML_DONE

# ----- MAIN CODE STARTS HERE ----- #

my $sql = "select distinct playlist_name from m3u";
$sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);

    my $even = $dir_cnt % 2; # flip flop the output to make 2 columns odd = left even =
right

    #make left right columns
    if(($even)) {
        print "<td align=center valign=top width=50%>";
    } else {
        print "<table align=center border=0 width=90% bordercolor=black name=outer>";
#start outer table
        print "<tr><td align=center valign=top width=50%>";
    }# end if

    print "<br><br><img src=http://$earhc/$image_url/$playlist_name/cover.jpg
align=center><br><br>";

    print "<table border=0 align=center width=90% bordercolor=black
name=internal_table>";

    print "<tr><td><input type=checkbox name=d$dir_cnt></td>";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >";
    print " --- Select All in Category ---</font>";
    print "</td></tr>";

    my $sql1 = "select pl_index,artist,song_name from m3u where playlist_name like
\"$playlist_name\"";
    # print "$sql1<br>";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();

    while (my $ref1 = $sth1->fetchrow_hashref()) {

```

```

    my $id = $ref1->{'pl_index'}; $artist = $ref1->{'artist'}; $song_name = $ref1-
>{'song_name'};
    $artist =~ s/_/_/g;
    $song_name =~ s/_/_/g;
    print "<tr><td><input type=checkbox name=\"\$id\"></td>\n";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$ song_name<br></font></td></tr>\n";
} #end while ref1

print "</table>"; # end inner table

if($seven) {
    print "</td></tr>";
    print "</table>"; #end outer table
    print "<center><br><font size=2 color=white face = arial,san serif, helvetica >";
    print "<a href=\"#button\">Return to top</a></font></center>";
} else {
    print "</td>";
} #end ending left right column if

$dir_cnt++; # increment counter used for left/right flip flop

} #end while ref -> display of play lists loop
#-----
print "<br><br>";

print "<input type=hidden name=size value=$dir_cnt>";

# $dbh->disconnect();

print "</form>";

} #end if (Submit)
#-- done displaying the selection list --

#-----#
# ----- DISPLAY RESULTS OF SELECTIONS ----- #
#
#-----#

if(defined(param("Submit.x")) || defined(param("Submit.x"))){
    my @play_list;
    my @list_copy;
    my @p_list;
    my $pl;
    my $first_pl;
    my $qry1;
    my $qry2;

```

```

# $num_of_dirs = param("size");

#
=====
#my $sql = "select distinct playlist_name from m3u order by playlist_name asc";
my $sql = "select distinct playlist_name from m3u";
    $sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);
} #end while "get all playlists" query
#
=====
# -----
# parse param list for "all in playlist" request = "d+some num" & individual songs "num"
# -----
my @song_list;
#my $pl_limit = 5;
#my $song_limit = 50;
#my $pl_count = 1;

foreach (param()) {
    push(@song_list,$_) if /^d+$/;    #just a song number
    # print "<br>$_" if /^d+$/;
    if (/^d\d+$/){
        #all in playlist d+number
        s/^d//;          #remove letter "d"

        push(@p_list,$play_list[$_]); #store playlist number
        push(@list_copy,$play_list[$_]);

    } # end if "d+numb all in playlist "
} # end foreach "item in param() list"
# -----
# Build "All in playlist" query string
# -----

if(@p_list){

    my $first_pl = shift @p_list;
    my @list_copy = $first_pl;
    $qry1 = "SELECT * FROM m3u WHERE playlist_name IN (\"$first_pl\"";

    foreach my $pl (@p_list){
        # last if($pl_count = $pl_limit);

```

```

    #print "<br>=". @p_list; #print "<br>@p_list";

    $qry1 .= ", \"$pl\" ";

    push(@list_copy,$pl);
    # $pl_count++;
  } #end foreach loop to build items in query string

  $qry1 .= ") order by playlist_name"; # query ends here

  #print "<br>$qry1";

} #end if anything in p_list array

# -----
# Build individually selected songs query string
# -----
if(@song_list){ #anyone home in song lists? go for it!

  my @song_copy = @song_list;

  my $first = shift @song_list;
  $qry2 = "SELECT * FROM m3u WHERE pl_index IN ( $first" ;
  foreach my $blarg (@song_list){
    #print "<br>=". @song_list;
    #print "<br>@song_list";
    $qry2 .= ", $blarg ";
  } #
  $qry2 .= ")";
  # $qry2 .= " limit $song_limit"; #add limit to number of songs
  #print "<br>$qry2";
} #end if anything in song_list array

# -----
# Now create m3u playlist file
# Execute query statement(s) and write file
# -----

#print "<br>here's your check of array size greater than zero
defined($qry1),defined($qry2)<br>";
if($qry1 || $qry2){
  # ---- Create unique play list name using numbers from date parts ----- #
  my $theDate = localtime;
  $theDate =~ /(\d+):(\d+):(\d+)\b/i;
  my $playlist_name = "Momi_playlist_$2$3.m3u";
  # ----- #

  print "<center><font size=2 color=white face = arial,san serif, helvetica >";

```

```

print "<b>Songs in your play list:</b></font></center>";
print " <table align=center>";

open(OUTFILE,">../htdocs/temp/$play_list_name");

print OUTFILE '#EXTM3U'."\n"; #write file header
#print '<br>#EXTM3U<br>'; #write file header

if($qry1){
  my $list = "";
  # now do the playlists
  my $sth1 = $dbh->prepare($qry1);
  $sth1->execute();
  while (my $ref1 = $sth1->fetchrow_hashref()) {
    my $playlist_name = $ref1->{'playlist_name'};
    my $pl_header = $ref1->{'pl_header'};
    my $song_url = $ref1->{'song_url'};
    $song_url =~ s/8080/8081/g;
    print OUTFILE "$pl_header\n$song_url\n";
  } # end while
  #print "<b>my list_copy array = @list_copy</b><br>";
  foreach my $list (@list_copy){
    $list =~ s/_/_/g;
    print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >All Songs
from <b>\"$list\"</b></font></td
></tr>\n";
  } # end foreach
}#end if $qry1 is defined

if($qry2){
  # now do the selected songs
  my $sth2 = $dbh->prepare($qry2);
  $sth2->execute();
  while (my $ref2 = $sth2->fetchrow_hashref()) {
    my $artist = $ref2->{'artist'};
    my $song_name = $ref2->{'song_name'};
    my $pl_header = $ref2->{'pl_header'};
    my $song_url = $ref2->{'song_url'};
    #print "$pl_header<br>$song_url<br>";
    $artist =~ s/_/_/g;
    $song_name =~ s/_/_/g;
    $song_url =~ s/8080/8081/g;
    print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$ song_name</font></td></tr>";
    print OUTFILE "$pl_header\n$song_url\n";
  } # end while
}# end if $qry2 is defined

close OUTFILE;

```

```

    print "</table>";
    print "<center><br><br><br>";
    print "<a href=\"http://www.themomi.org/temp/$play_list_name\">";
    print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
    print "</font>";
    print "<br><br><br>";

# ----- include text for nubies ----- #

print "<br><br>";
print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the button above to launch
your playlist<br> with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>

<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the button above and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>

<tr><td colspan=3 align=center>
MORE

#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>

print "</table><br><br>";

}else{
    print "<br><b>Nothing Selected</b><br>";

    #foreach (param()) {
    #    print "$_<br>";
    #}#end foreach
    #print "The value of Param(\"Submit.x\") is " , param("Submit.x") , "<br>";

```

```
#print "The value of defined Param(\"Submit.x\") is " , defined(param("Submit.x")) ,  
"<br>;
```

```
$dbh->disconnect();
```

```
}#end if defined queries  
}# end if (Submit.x) "display results of selection "
```

```
print "</body>";  
print end_html();
```

```
#bye bye! Thanks for visiting!
```

```
exit(0);
```

```
1;
```

Alternatively, at step 3-410, the received media content play list may include a random media content delivery choice that the user of client device 3-204 may select to transmit a request to web server 3-210 for delivery of random media content, wherein the user ID automatically accompanies the request. An embodiment for implementing the delivery of media content in a random manner to client computer 3-204 is described in detail within reference to step 3-424.

At step 3-412 of Figure 3-4A, the present embodiment causes web server 3-210 to determine whether the requesting media application operating on client computer 3-204 is a valid media application. One of the main functions of a valid media application may be that it is a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If the web server 3-210 determines at step 3-412 that the requesting media application operating on the request client computer 3-204 is a valid media

application, the present embodiment proceeds to step 3-416. However, if the web server 3-210 determines at step 3-412 that the requesting media application is not a valid application, the present embodiment proceeds to step 3-414.

In step 3-414, the present embodiment causes web server 3-210 to redirect client computer 3-204 so that it is prevented from accessing the source of the media content. It is understood that the redirection of step 3-414 may be performed in diverse ways. For example, the present embodiment may cause web server 3-210 to redirect client computer 3-204 to a software program that identifies it, logs it out of web server 3-210 and prevents any subsequent logging in of it for a defined amount of time (e.g., 10 minutes, an hour, a day, a month, a year, or any defined amount of time).

At step 3-416, the present embodiment causes the web server 3-210 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer 3-204 is valid. If the web server 3-210 determines that the user ID is not valid at step 3-416, the present embodiment proceeds to step 3-414. However, if the web server 3-210 determines that the user ID is valid at step 3-416, the present embodiment proceeds to step 3-418. It is noted that the order in which steps 3-412 and 3-416 are performed by the present embodiment may be switched from the order illustrated within Figure 3-4A. That is, the present embodiment may perform step 3-416 before performing step 3-412.

In step 3-418 of Figure 3-4A, the present embodiment causes web server 3-210 to transmit to client computer 3-204 a redirection command along with a

time sensitive access key (e.g., for that day, hour, or for any defined timeframe) thereby enabling the client to eventually receive the requested media content. The redirection command may include a time sensitive address of where the media content is location within content server 3-212. The address is time sensitive because periodically the present embodiment of content server 3-212 renames some or all of the media address directories thereby making previous content source addresses obsolete. It is noted that the actual location of the media content may not change within content server 3-212, just the address of the content is changed. However, the locations of the media content may actually be changed along with the corresponding addresses. In any case, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 3-212. As such, if someone with inappropriate intentions is able to find out where the media content source is stored, the next time he or she tries to use that route (or address) it does not exist anymore thereby preventing further unauthorized access.

It is noted that within an embodiment of the present invention, the addresses (or routes) of content server 3-212 that are actively coupled to one or more client devices (e.g., 3-204 to 3-208) are maintained while future addresses (or routes) are being created for new client devices. It is appreciated that as client devices are uncoupled from the media content source of content server 3-212, that directory address (or link) may be immediately changed thereby restricting access to unauthorized client devices or applications.

Alternatively, the functionality of redirection to a media content source may be implemented by utilizing a server network where multiple servers are content providers (e.g., 3-212) or by routing a requesting client device (e.g., 3-204, 3-206 or 3-208) through multiple servers. Within another embodiment in accordance with the present invention, the delivery of media content from a central content provider (e.g., 3-212) may be routed through one or more intermediate servers before being received by the requesting client device (e.g., 3-204, 3-206 or 3-208).

The redirection functionality of step 3-418 is well suited to be implemented in a wide variety of ways. The following Perl Script entitled "Redirector Program" includes an exemplary implementation for the redirection functionality of step 3-418 along with other functionality of flowchart 3-400. Additionally, this Perl Script records the Internet Protocol (IP) addresses of the client computers (e.g., 3-204), the media content requested and its transfer size. In this manner, an implementation of flowchart 3-400 is able to accurately meter royalty payments, clock usage and transfers, and also keep a log of media content popularity.

```
#!/usr/bin/perl -w
```

```
#-----
#           Redirector Program
#-----
# This program obscures the proper address of the mp3 source file for due diligence
# purposes and records the requested mp3 file name, requestor ip number, file size
# and date of request to allow for accurate royalty payments & bandwidth monitoring,
# The resulting data is stored in a .csv file for ease of subsequent analysis.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
#----- #
```

```

my $version = 4.0;

# -----

$SIG{CHLD} = "IGNORE"; #kill zombies! Oooo scary scary!;
use IO::Socket::INET;
use DBI();

my $debug = 0; #debugging messages on/off

##### localize these variables #####
#
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
#   choices are: changing all the time
# real_dest_url: location of the content server
# send_away_url: location that browsers are sent to if the try to access mp3s.
#
#
#program name
my $level      = "hi";      # hi , mid, lo, x_lo;
my $short_name = $level . "_redir.cgi";
my $program_name = "http://www.themomi.org/redirs/$short_name";

# logs
my $apache_log = "/usr/local/apache/logs/access_$level.log";
my $error_log  = "/usr/local/apache/logs/error_$level.log";

#port
my $server_port = 8081;

#urls
my $real_dest_url = "www.mpb.tv";
my $send_away_url = "www.themomi.org";

#db connection
my $db_host = "AAA.BBB.CCC.DD";
my $db_user = "my user name";
my $db_pw   = "some password";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "$db_user", "$db_pw",
                      { 'RaiseError' => 1 }) or die "can't connect to db";

#rate control songs/min limit
my $limit = 10;

# Changeable access parameters read in from database
my $mp3_root_dir;
my $pass_key;

# change content directory names to prevent direct linking

my $sql_db = "Select dir from file_location ";
$sql_db .= "where band like \"\$level\"";

```

```

my $sth_db = $dbh->prepare($sql_db);
    $sth_db->execute();
my $ref_db = $sth_db->fetchrow_hashref();

$mp3_root_dir = $ref_db->{'dir'};
    $sth_db->finish();

if(!defined($mp3_root_dir)) {
print "oops!\n";
#-----
# $mp3_root_dir = "hi_band";
    $mp3_root_dir = "Fatty";
# $mp3_root_dir = "ookii";
#-----
# $mp3_root_dir = "mid_band";
# $mp3_root_dir = "chu";
# $mp3_root_dir = "norman";
#-----
# $mp3_root_dir = "lo_band";
# $mp3_root_dir = "chisai";
# $mp3_root_dir = "Slim_Jim";
#-----
# $mp3_root_dir = "x_lo_band";
# $mp3_root_dir = "chibi";
# $mp3_root_dir = "Ally_McBeal";
#-----

} #end if

my $sql_key = "Select cache_key from cache_key ";
my $sth_key = $dbh->prepare($sql_key);
    $sth_key->execute();
my $ref_key = $sth_key->fetchrow_hashref();

$pass_key = $ref_key->{'cache_key'};
    $sth_key->finish();

#####
my $msg = "debug mode off";
while ($_ = $ARGV[0]) {
    shift;
    #print "found $_ \n";
    last if /^-$/;
    if (/^D/i || /^X/i || /^V/i) {
        $msg = "debug mode on";
        $debug = 1;
    }
}
}#end while

print "\nStarted program: $short_name Version: $version at Port: $server_port $msg \n";
print " Using $real_dest_url/$mp3_root_dir as music source dir\n";

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                                Type    => SOCK_STREAM,
                                Reuse   => 1,
                                Listen  => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

```

```
#####

# initialize variables used for player discrimination
my $ok = 1; # on by default;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0; # this is the winamp player
my $real = 0; # this is the real player
my $ripper = 0; # this is for stream ripper etc.
my $speeder = 0; # this is for speeders
my $what = "";
my $whole_request = "";
my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

# ----- #
# Now do main loop. fork off child when connection made
# ----- #

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here

        my $client_info = getpeername($client);
        (my $port, my $iaddr) = unpack_sockaddr_in($client_info);
        my $remote_host_ip = inet_ntoa($iaddr);
        my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
        my $date = localtime;

        if(! defined($remote_hostname) ){ $remote_hostname = "unknown";}
        if($remote_host_ip =~ /10.10.0.252/o || $remote_host_ip =~ /10.10.0.251/o){
            exit(1);
        }

        if($debug){
            print "\n ----- \n";
            print "$short_name: client ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . "
\n";
        }

        #evaluate the first line
        $what = <$client>;
        #ignore network health check from x.x.x.104
        if(! defined($what) && $remote_host_ip !~ /209.10.35.104/){
            print "$short_name@$server_port encountered a client who said nothing! $remote_host_ip ,
$date<-- \n "
;
            exit(1);
        }
    }
}
```

```

    }
    $first_line = $what;

    if($debug){ print "\n First line is \"", $what, "\"\n"; }
    if ($what !~ /^(GET|HEAD)\s+/i) { #valid requests must start with GET or HEAD
        if($remote_host_ip !~ /209.10.35.104/o){
            print "$short_name -> bad req HTTP/1.1 400, no GET HEAD, $remote_host_ip said:->$what<- \n";
        }
        print $client "HTTP/1.1 400, \"Bad Request\"";
        exit(1);
    }
    if ($what =~ /$mp3_root_dir/i) {
        print "$short_name -> bad req HTTP/1.1 400, mp3_dir in request, $remote_host_ip said:->$what<- \n";
        print $client "HTTP/1.1 400, \"Bad Request\"";
        exit(1);
    }
}

#banned ip list

if ($remote_host_ip =~ /132.239.12.77/
    || $remote_host_ip =~ /66.122.240.11/
    || $remote_host_ip =~ /63.17.233.236/) {
    #print " $short_name sending HTTP/1.1 400, baned $remote_host_ip \n said:->$what<- \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

#now parse line, extract what we need or dump 'em
if ($what =~ /^(([A-Z]+)\s+([\S]+)\s+HTTPV([\d\.]+)\s*)/){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;
}else{
    print " $short_name: sending bad client HTTP/1.1 400, Bad Request message\n";
    print " $short_name: bad req was: $what\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

# print "method is > $what_method< song is >$what_song< version is >$what_version< \n";

if ($what_method !~ /(GET|HEAD)/i) {
    exit(1);
}

#now loop to get other info lines
while( ($request = <$client>) ne ("r\n" || "r\nr\n") ){
    last if(! defined($request));
    $request =~ s/[r\n]//g;
    if($debug) {
        print "<$level: $counter >The client said: " . $request . "<\n";
    }
    #----- WEB BROWSER INFO ----- #
    if($request =~ /Mozilla/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /Opera/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }

```

```

    }
    if($request =~ /Omniweb/io) {
        $ok = 0; $muzzy = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    if($request =~ /iCab/io) {
        $ok = 0; $muzzy = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- MP3 RIPPER INFO ----- #
    if($request =~ /StreamRipper/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- No get for Wget ----- #
    if($request =~ /Wget/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- FlashGet fizzles ----- #
    if($request =~ /FlashGet/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- GetRight is Wrong ----- #
    if($request =~ /GetRight/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- Step on Spiders ----- #
    if($request =~ /asterias/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- Web access = No access ----- #
    if($request =~ /Web access/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- Monica gets the bone ----- #
    if($request =~ /monica/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- man what the hell is this? ;- ) ----- #
    if($request =~ /Media Jukebox WinInet Reader/io){
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- burn Nero Player ----- #
    if($request =~ /NeroMediaPlayer/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- 1SMUSIC is deaf ----- #
    if($request =~ /1SMUSIC/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- I know who Anonymizer is ----- #
    if($request =~ /Anonymizer/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    #----- MP3 CLIENT INFO ----- #
    if($request =~ /Windows-Media-Player/io){
        $ok = 0; $nsplayer = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }
    if($request =~ /NSPlayer/io) {
        $ok = 0; $nsplayer = 1; $request =~ /\.+:s*(.+)/; $ua = $1;
    }

```

```

    }
    if($request =~ /RMA/io) {
        $ok = 0; $real = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;
    }
    if($request =~ /Winamp/io) { $ok = 1; $request =~ /\.+:\s*(.+)\s*$/; $ua = $1; }
    if($request =~ /QuickTime/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /Xaudio/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /AppleApp/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /iTunes/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /Sonique/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /xmms/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }
    if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1; }

    #=== hold for error reportin ===#
    if($request =~ /User Agent/io) { $u_a_line = $request; }

    last if ($counter > 20 ); #we don't want some client babbling on and on
    $counter++;

}#end while

if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
}

# ##### Rate controls to prevent excessive access to content

$remote_host_ip =~ /\.+\.+/;
# my $last_octet_mod = $1;
# $last_octet_mod = $last_octet_mod % 1; #mod by number of tables we want
# my $time_offset = 1;

my $sql0 = "Select count(ip_address) as loser_count from loser ";
$sql0 .= "where ip_address = \"\$remote_host_ip\" ";
$sql0 .= "AND user_agent = \"\$ua\" ";
my $sth0 = $dbh->prepare($sql0);
$sth0->execute();
my $ref0 = $sth0->fetchrow_hashref();
my $loser_count = $ref0->{ 'loser_count' };
$sth0->finish();

##
if($loser_count <= 0){

my $sql = "SELECT COUNT(request_time) as number_songs_reqs ";
# $sql .= "MINUTE(MIN(request_time)) - MINUTE(now()) as duration ";
$sql .= "FROM rate_control_0 ";
$sql .= "WHERE ";
$sql .= "ip_address = \"\$remote_host_ip\" ";
$sql .= "AND ";
$sql .= "user_agent = \"\$ua\" ";
$sql .= "AND ";
$sql .= "request_time > now() - INTERVAL 60 SECOND";

print "\n$sql\n" if $debug;

my $sth = $dbh->prepare($sql);
$sth->execute();

```

```

my $ref = $sth->fetchrow_hashref();
my $song_count = $ref->{'number_songs_reqs'};
#my $duration = ($ref->{'duration'});
$sth->finish();

print "\n Song count is $song_count \n" if $debug;

if($song_count >= $limit){
    $speeder = 1;
    $ok = 0;
    print "\n Loser exceeded limit\n" if $debug ;

    my $sql1 = "insert into loser (ip_address,user_agent,loser_time) values
(\$remote_host_ip\", \"\$ua\", now()
)";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if song_count

}else{
    print "\n loser is in loser table\n" if $debug;
    $speeder = 1 ;
    $ok = 0;
}

if(!$speeder){
    my $sql1 = "INSERT INTO rate_control_0 ";
    $sql1 .= "(ip_address,user_agent,request_time) ";
    $sql1 .= "VALUES (\$remote_host_ip\", \"\$ua\", now())";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if

print $client "HTTP/$what_version 302 Found \r\n";
print " I told him HTTP/$what_version 302 Found \n" if $debug;
print $client "Date: $date \r\n";
print $client "Server: Apache/1.3.20 \r\n"; #lie to 'em to keep 'em guessing.
print $client "Keep-Alive: timeout=10, max=20\r\n";

#----- Good guys ----- #
if(($nsplayer || $real) && ! $speeder){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    print "found nsplayer or real player\n" if $debug;
}
if($ok && ! $speeder){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($mozzzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
}

```

```

# print "mozzy type web server being redirected to index page\n";
}
if($ripper){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
# print "mozzy type web server being redirected to index page\n";
}
if( $speeder ){
    print "$short_name: Speeder @ $remote_host_ip sending bad client HTTP/1.1 400, Bad Request
message\n";
    print "$short_name: Speeder said $what\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
}

$ok = 1;
$nsplayer = 0;
$mozzy = 0;
$real = 0;

print $client "\r\n\r\n";
# print "\ndone\n";
# print "*****\n\n";

# $dbh->disconnect();
close $client;

$what_song =~ /(.(+)\.mp3)/io;
$what_list = $1;
$song_name = $2;

#trash filter *****
if(! defined($what_list)){
    print "$short_name: no list\n"; $what_list = "none\n";
}
if(! defined($song_name)){
    print "$short_name: no song_name $remote_host_ip\n"; $song_name = "none";
    $bad = 1;
}
if(! defined($remote_hostname)){
    print "$short_name: no hostname\n"; $remote_hostname = "none";
    $bad = 1;
}
if(! defined($remote_host_ip)){
    print "$short_name: no host_ip\n"; $remote_host_ip = "none";
    $bad = 1;
}
if(! defined($remote_hostname)){
    print "$short_name: no hostname $remote_host_ip\n"; $remote_hostname = "none";
    $bad = 1;
}
if(! defined($ua)){
    print "$short_name: no ua\n"; $ua = "none";
    $bad = 1;
}

#match this: APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-
agent}i\")
#change date from: Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

```

```

$date =~ /.+\s(.+)\s(.+)\s(.+)\s(.+)/io;
my $month_name = "$1";
my $day = "$2";
my $time = "$3";
my $year = "$4";

if ($day < 10){ $day = "0$day"; } #hack to add leading zero

$month_name =~ s/ //g;
#my $apache_date = "$day/$1/$4:$3 -0500";
my $apache_date = "$day/$month_name/$year:$time -0500";
$whole_request =~ s/\r//;
$whole_request =~ s/\n//;

open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
$a_str = "$remote_host_ip - - $apache_date \"$whole_request\" 302 2 \"$program_name\" \"$ua\"";
print APACHE_LOG "$a_str\n";
close APACHE_LOG;

if($bad || $ua eq "unknown" ){
    $first_line =~ s/[\r\n]//g;
    open (ERROR_LOG, ">>$error_log");
    my $e_str = "$remote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
    print ERROR_LOG "$e_str\n";
    close ERROR_LOG;
} #end if error

close STDOUT;
close STDERR;
close STDIN;

exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
$dbh->disconnect();
close ($server);
exit(0);

```

At step 3-420 of Figure 3-4B, upon receiving the redirection command, the present embodiment causes the media application operating on client computer 3-204 to automatically transmit to content server 3-212 a new media delivery request (that automatically includes the time sensitive access key) that includes the address of the desired media content.

In step 3-422, the present embodiment causes content server 3-212 to determine whether the time sensitive access key associated with the new media delivery request is valid. If the content server 3-212 determines that the time sensitive access key is not valid at step 3-422, the present embodiment proceeds to step 3-414 of Figure 3-4A. However, if the content server 3-212 determines that the time sensitive access key is valid at step 3-422, the present embodiment proceeds to step 3-424.

At step 3-424 of Figure 3-4B, the present embodiment causes content server 3-212 to transmit the requested high fidelity media content to client computer 3-204. It is noted that when content server 3-212 transmits the media content to client computer 3-204 it does not encrypt the data stream. Additionally, content server 3-212 transmits the media content in a burst load (rather than a fixed data rate) that transfers to client device 3-204 as fast as its network transfer rate allows. The present embodiment may cause content server 3-212 to adapt its download speed to whatever network transfer rate the client computer's is able to handle. For example, if client computer 3-204 is coupled to the Internet 3-202 via a T1 communications line, content server 3-212 transfers the media content to client computer 3-204 at transmission speeds allowed by the T1 communications line. As such, once the requested high fidelity media content is transmitted to client computer 3-204, content server 3-212 is then able to transmit requested high fidelity media content to another client computer (e.g., 3-206 or 3-208). It should be appreciated that by performing step 3-424 in this manner, the delivery of high fidelity media content by the present embodiment becomes very efficient in

terms of a statistical distribution over time and it does not overly congest the utilized communication network(s).

It is appreciated that the delivery of the media content to client device 3-204 by content server 3-212 at step 3-424 may be implemented in a wide variety of ways. For example, while delivering the media content to client computer 3-204 at step 3-424, a rate control restrictor functionality may be resident to content server 3-212 in order to monitor and limit "suspicious" media content retrieval. For instance, if a client device (e.g., 3-204) tries to retrieve media content from content server 3-212 faster than a predefined limit, the rate control restrictor uncouples client device 3-204 from its media content source (e.g., content server 3-212) and continues to restrict its access from the source for a predefined amount of time, as described herein. One of the reasons for implementing this rate control restrictor is to restrict access to those unauthorized applications that are designed to retrieve and/or copy media content from the source in an unauthorized manner. For example, a client device (e.g., 3-204) may be retrieving pieces of audio content at a rate determined to be much faster than is humanly possible to listen to in real time. As such, the rate control restrictor (which may be implemented with software and/or hardware) uncouples the client device from the media content source (e.g., content server 3-212) and restricts its access for a predefined amount of time.

Alternatively, an embodiment of the present invention may be implemented such that a user of a client device (e.g., 3-204) may establish a huge buffer of media content (e.g., audio clips, video clips, and the like) to be delivered by

content server 3-212. However, the present embodiment at step 3-424 may just provide client computer 3-204 one piece of media content at a time as would be typically needed to utilize (e.g., listen to, watch, etc.) that piece of media content in real time.

Within another embodiment in accordance with the present invention, embedded keys and/or digital watermarks may be implemented within media content stored by content server 3-212 which may be delivered during flowchart 3-400 of Figures 3-4A and 3-4B. By using embedded keys and/or digital watermarks within the media content, it is easier to determine if some unauthorized person has been retrieving and/or copying media content from a media content source of content server 3-212. The embedded keys and/or digital watermarks within media content may include, but are not limited to, information indicating where the media came from, the identity of the media requestor and/or the identity of the requestor client device (e.g., 3-204, 3-206 or 3-208). With this information, it may be determined by the present embodiment who or what client device has been unauthorized in retrieving and/or copying media content from one or more of the media sources (e.g., 3-212). As such, that person and/or client device (e.g., 3-204, 3-206 or 3-208) can be permanently restricted by the present embodiment from requesting and/or accessing media content.

It is understood that if the user of client computer 3-204 caused it at step 3-410 to transmit to web server 3-210 a request for delivery of random media content, this request information may be communicated to content server 3-212 in order for it to fulfill the request at step 3-424. The delivery of media content in a

random manner at step 3-424 may be implemented in diverse ways in accordance with the present embodiment. For example, a random media play list may be generated by first taking all of the titles of the media content stored by content server 3-212 and concatenated them into a single dimensioned array. Next, a random number may be generated. It is appreciated that a random number may be generated in a wide variety of ways in accordance with the present embodiment. For instance, a random number generator may incorporate as its product the combination of:

1. A number produced by a random function. It is noted that the random number may not truly be a random number, but a random sequence that follows a "seeded" number. This seeded number is a starting point for the further permutations of random selection, but would have been predictable with a fixed seed value.

2. The Julian date maintained within content server 3-212 that includes the hours, minutes, seconds, day, date, and four-digit year.

Using this combination, the following code may then be executed:

```
$theDate = 'date'; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "MoMI_random_playlist_$2$3.m3u";

$random_seed = "$3$2$3"; #use time parts to make random
srand $random_seed; #seed the random num gen.
```

It is appreciated that the time/date provides a basis for seeding the random number generator that is repeatable once per millennium. Now that a random number has been selected, the actual process of applying this to the array of media content titles follows.

Exemplary criteria for selecting a media content title for inclusion in the random play lists may be, but is not limited to, three-fold:

1. There cannot be more than 50 media content titles total in the play list;
2. The selection should come from no more than 20 play lists stored by content server 3-212; and
3. The selection should not have been previously selected.

Assuming that the first two conditions have not been met, the resulting random number is used as the index into the media content title array and selects the song title that resides in that location in the array. If that title has been previously selected, a new random number is requested, and the process continues until all 3 conditions are fulfilled.

The resulting list of media titles are then formatted into a useable play list (or what may be referred to as a M3U file), and it is written to a memory device of content server 3-212 while a page is prepared for presenting this information to the user of client computer 3-204.

The following Perl Script entitled "Random Play List Program" is an exemplary implementation for enabling content server 3-212 to provide a random play list of media content as described herein.

```
#!/usr/bin/perl -w  
#-----  
#
```

```

#                RANDOM PLAY LIST PROGRAM
#
# This program reads in a mp3 play list source file in many directories ,
# displays the random choice of file names ,
# and creates a new file to download.
# Programmer: Ted Fitzgerald
# Created for "themomi.org"
#-----
my $version;
$version = 5.0;
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# declare variables

my ($i,$j); #just counters
my ($play_list_name); #variable to create random file name for download
my ($show_many_songs); # = total number of songs in all the dirs.
my (@thename); #array to contain the name of the randomly selected song
my (@thepath); #array to contain the URL path of the randomly selected song
my ($showname); #used to find the portion of name we want to display.
my ($song_url);
my (@theheader);
my (@artist);
my ($artist);
my ($theDate);
# ----- user ajustable paramerters -----
my $max_num_songs = 34; # set limit of how many songs to display
my $maxdirs = 20; #set limit on number of dirs to look in
my $mp3dir = "../htdocs/mp3"; #path from cgi script to mp3 directory
my $center_col_size = ($max_num_songs /2 + 1);

$|=1; #flush the buffer
use strict;
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "PIZZA_MAN", "HE_DELIVERS",
                      { 'RaiseError' => 1 }) or die "can't connect to db";

my $q = new CGI;

print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>Random Play List Creation</title>";

```

```

print "<style><!--a{ text-decoration:none }!--></style>";
print "<style><!--a:hover{color:#3399cc; }--></style>";

print <<HTML_DONE;
<SCRIPT LANGUAGE="JavaScript">
<!--

function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}

// -->
</SCRIPT>
<head>
<body bgcolor="black" topmargin="0" leftmargin="0" marginheight="0"
marginwidth="2" text="white" link="#ffcc66" vlink="#ffcc
66" alink="#ffcc66">

<font size=2 color=white face = arial,san serif, helvetica >
HTML_DONE

$theDate = localtime; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "Momi_random_playlist_$3$2$3.m3u";
#-----#
# do all file io before displaying the names in list

my $sql = "select pl_header,song_name,song_url,artist from m3u order by rand() limit
$max_num_songs";

my $sth = $dbh->prepare($sql);
    $sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $pl_header = $ref->{'pl_header'};
    my $song_name = $ref->{'song_name'};
    my $song_url = $ref->{'song_url'};
    my $artist = $ref->{'artist'};
    push (@thename,$song_name);
    push (@thepath,$song_url);
    push (@theheader,$pl_header);
    push (@artist,$artist);
}

open(OUTFILE,">../htdocs/temp/$play_list_name");

```

```

#write out file header
print OUTFILE "#EXTM3U\n";
for($i=0;$i<$max_num_songs;$i++){

    print OUTFILE "$theheader[$i]\n";
    $song_url = "$thepath[$i]";
    $song_url =~ s/8080/8081/io;
    print OUTFILE "$song_url\n";

}#end for
close OUTFILE;

#-----#
#now show 'em what they have in the list

print<<MORE;

<center>
MORE
print "<br>";
print "</center>";
print "<table align='center' border=0 width = \"95%\">";
print "<tr width=45%><td><td rowspan=$center_col_size align=center valign=top>";

print <<HERE;
<A HREF="http://www.themomi.org/temp/$play_list_name"
    ONMOUSEOVER="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert_over.gif');
return true;"
    ONMOUSEOUT="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert.gif'); return
true;">
    <IMG NAME="create_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/create_now_vert.gif"
BORDER=0></A>
HERE

print "</td>";
print "<td width=45%></td></tr>";
for($i=0;$i<$max_num_songs;$i++){
    my $flip_flop = ($i % 2 );

    $showname = $thename[$i];
    # $showname =~ /EXTINF:\d+,(.+)/i;
    $showname =~ s/_/ /g;
    $artist = $artist[$i];
    $artist =~ s/_/ /g;

    if($flip_flop){

```

```

    print "<td width = 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showname
</font></td></tr>";
    }else{
        print "<tr><td width= 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showna
me </font></td>";
    }

}
print "</table><br>";

#-----#

# print "<center>";
# print "<br><a href=\"http://www.themomi.org/temp/$play_list_name\">";
# print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
# print "</center>";
    print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>&nbsp;</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">
MORE

print "<A HREF=\"http://www.themomi.org/perl/random_hi.cgi\"";

print<<MORE;
    ONMOUSEOVER="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm_over.gif');
    return true;"
    ONMOUSEOUT="changeImages('rerandom_now_vert_01',
'http://www.themomi.org/museum/newimages/re-randomize_sm.gif');
    return true;">
        <IMG NAME="rerandom_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/re-randomize_sm.gif"
BORDER=0></A>
</font></td>
<td>&nbsp;</td>
</tr>
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the center button above to launch
your <br>playlist with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>

```

```
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
```

```
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the center button and
use "save target as" <br> to save your playlist file to your computer.</font>
```

```
</td></tr>
```

```
<tr><td colspan=3 align=center>
```

```
MORE
```

```
#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
```

```
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
```

```
#play the songs from MPB.TV whenever you wish</font><br>
```

```
#</td></tr>
```

```
print "</table><br><br>";
```

```
print "<center>";
```

```
print $q->end_html;
```

```
1;
```

In step 3-426 of Figure 3-4B, upon receiving the requested high fidelity media content from the content server 3-212, the present embodiment causes client computer 3-204 to store it. The present embodiment may cause client device 3-204 to store the received media content in a manner such that it is not easy for the user of client computer 3-204 to redistribute the media content in an unauthorized manner. For example, the present embodiment may cause the high fidelity media content to be stored by a memory device of client device 3-204 utilizing one or more hidden directories where it may be cached for a limited amount of time. As part of restricting access to the media content, the present embodiment at step 3-412 may reject any media player application as invalid if it allows accessibility to the hidden directory. It is noted that if the user of client computer 3-204 turns it off or quits out of the media player application, the stored media content is typically deleted from the memory of client 3-204.

At step 3-428, the user of client computer 3-204 may cause the media application operating on it to access and utilize the delivered high fidelity media content thereby enabling its user or users to experience (e.g., listen to, view, etc.) the media content. It is noted that a specialized or proprietary media player is not needed in order to utilize the received media content. Instead, an industry standard media player may be utilized by client computer 3-204 to utilize the received media content. Some of the exemplary industry standard media players that are typically available at no cost and are supported by longstanding organizations may include, but are not limited to, Windows™ Media Player™ for personal computers (PCs), iTunes™ Player or Quicktime™ for Apple computers, and XMMS Player for computers utilizing the Linux operating system.

Accordingly, the present invention provides a method and system for delivering “on-demand” high fidelity music to computer systems via the Internet which does not involve proprietary audio players and/or encryption of the music. Additionally, the present invention provides a method and system that includes the above accomplishment and does not overly congest the communication networks of the Internet. Furthermore, the present invention provides a method and system that includes the above accomplishments and monitors the music (or media) delivered in order to compensate the owner of copyrighted music (or media) for it.

One embodiment of the present invention enables delivery of “on-demand” high fidelity media content to computers via the Internet while restricting unauthorized users from directly retrieving media content from its source

database. Once the computer receives the media, it is stored using hidden directories so that it may not be easily shared with others. Within the present embodiment, there are different functionality that are implemented in order to protect and monitor the media content source. For example, the actual address location of the media database is hidden from content recipients while its address directory is periodically change making past addresses obsolete. Additionally, an access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media content requests. By implementing these and other functionality, the present embodiment restricts redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 4: SYSTEM AND METHOD FOR PROVIDING GLOBAL MEDIA
CONTENT DELIVERY

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital system memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical

quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that throughout discussions of the present invention, discussions utilizing terms such as “allowing” or “preventing” or “restricting” or “utilizing” or “regulating” or “providing” or “determining” or “controlling” or “transmitting” or “receiving” or “recognizing” or “generating” or “storing” or the like, refer to the action and processes of a computing system, or similar electronic computing device, that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system’s registers and memories and is transformed into other data similarly represented as physical quantities within the computing system memories or registers or other such information storage, transmission, or display devices.

Figure 4-1 is a block diagram of an embodiment of an exemplary computer system 4-100 that may be used in accordance with the present invention. It is

understood that system 4-100 is not strictly limited to be a computer system. As such, system 4-100 of the present embodiment is well suited to be any type of computing device (e.g., server computer, desktop computer, laptop computer, portable computing device, etc.). Within the discussions of the present invention, certain processes and steps are discussed that are realized, in one embodiment, as a series of instructions (e.g., software program) that reside within computer readable memory units of computer system 4-100 and executed by a processor(s) of system 4-100. When executed, the instructions cause computer 4-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 4-100 of Figure 4-1 comprises an address/data bus 4-110 for communicating information, one or more central processors 4-102 coupled with bus 4-110 for processing information and instructions. Central processor unit(s) 4-102 may be a microprocessor or any other type of processor. The computer 4-100 also includes data storage features such as a computer usable volatile memory unit 4-104, e.g., random access memory (RAM), static RAM, dynamic RAM, etc., coupled with bus 4-110 for storing information and instructions for central processor(s) 4-102, a computer usable non-volatile memory unit 4-106, e.g., read only memory (ROM), programmable ROM, flash memory, erasable programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), etc., coupled with bus 4-110 for storing static information and instructions for processor(s) 4-102.

System 4-100 also includes one or more signal generating and receiving devices 4-108 coupled with bus 4-110 for enabling system 4-100 to interface with other electronic devices. The communication interface(s) 4-108 of the present embodiment may include wired and/or wireless communication technology. For example, in one embodiment of the present invention, the communication interface 4-108 is a serial communication port, but could also alternatively be any of a number of well known communication standards and protocols, e.g., a Universal Serial Bus (USB), an Ethernet adapter, a FireWire (IEEE 1394) interface, a parallel port, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In another embodiment a digital subscriber line (DSL) connection may be employed. In such a case the communication interface(s) 4-108 may include a DSL modem. In any case, the communication interface(s) 4-108 may provide a communication interface to the Internet.

Optionally, computer system 4-100 can include an alphanumeric input device 4-114 including alphanumeric and function keys coupled to the bus 4-110 for communicating information and command selections to the central processor(s) 4-102. The computer 4-100 can also include an optional cursor control or cursor directing device 4-116 coupled to the bus 4-110 for communicating user input information and command selections to the central processor(s) 4-102. The cursor directing device 4-116 can be implemented using a number of well known devices such as a mouse, a track ball, a track pad, an optical tracking device, a touch screen, etc. Alternatively, it is appreciated that a cursor can be directed and/or

activated via input from the alphanumeric input device 4-114 using special keys and key sequence commands. The present embodiment is also well suited to directing a cursor by other means such as, for example, voice commands.

The system 4-100 of Figure 4-1 can also include a computer usable mass data storage device 4-118 such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) coupled with bus 4-110 for storing information and instructions. An optional display device 4-112 is coupled to bus 4-110 of system 4-100 for displaying video and/or graphics. It should be appreciated that optional display device 4-112 may be a cathode ray tube (CRT), flat panel liquid crystal display (LCD), field emission display (FED), plasma display or any other display device suitable for displaying video and/or graphic images and alphanumeric characters recognizable to a user.

Figure 4-2 is a block diagram of an exemplary network 4-200 that may be used in accordance with an embodiment of the present invention. For example, network 4-200 of the present embodiment enables one or more client computer systems (e.g., 4-204, 4-206 and/or 4-208) to receive high fidelity media content "on-demand" from a media content server 4-212 via the Internet 4-202 while restricting unauthorized users from directly retrieving media content from its source database stored by content server 4-212.

Network 4-200 of Figure 4-2 includes web application server 4-210 and content server 4-212 which are communicatively coupled to the Internet 4-202.

Additionally, web application server 4-210 and content server 4-212 may be communicatively coupled together within out utilizing Internet 4-202. Furthermore, network 4-200 includes client computers 4-204, 4-206 and 4-208 which are each communicatively coupled to the Internet 4-202. In this manner, web server 4-210, content server 4-212, and client computers 4-204 to 4-208 are able to communicate. It should be appreciated that the devices of network 4-200 of the present embodiment are well suited to be coupled in a wide variety of implementations. For example, content server 4-212, web server 4-210, and client computers 4-204 to 4-208 of network 4-200 may be coupled via wired communication technology (e.g., coaxial cable, copper wire, fiber optics, and the like) and/or wireless communication technology.

Within network 4-200, it is understood that web server 4-210, content server 4-212 and client computers 4-204 to 4-208 may be implemented in a variety ways in accordance with the present embodiment. For example, web server 4-210, content server 4-212 and client computers 4-204 to 4-208 may be implemented in a manner similar to computer system 4-100 of Figure 4-1. However, these devices of network 4-200 are not strictly limited to such implementation. Additionally, web server 4-210 and content server 4-212 of the present embodiment perform a variety of functionality within network 4-200. It is understood that web server 4-210 and content server 4-212 may actually reside on a single physical computing device (e.g., computer 4-100 of Figure 4-1). However, web server 4-210 and content server 4-212 of the present embodiment may each be implemented as one or more physical computing devices.

It is appreciated that network 4-200 of Figure 4-2 is able to operate with and provide delivery of any type of media content (e.g., audio, video, multimedia, graphics, information, data, software programs, and/or the like) in any type of format. For example, the content server 4-212 may provide audio clips, video clips, and the like to client computers 4-204 to 4-208 via the Internet 4-202.

Figure 4-3 is a block diagram of an exemplary system 4-300 for implementing access keys in accordance with an embodiment of the present invention. Specifically, system 4-300 illustrates a client computer system (e.g., 4-204) utilizing access keys in order to receive high fidelity media content "on-demand" from a media content server (e.g., 4-212) while unauthorized users or media applications are restricted from directly retrieving and/or copying media content from its media source database (e.g., 4-304).

The client computer device 4-204 may communicatively couple with a web/application server (e.g., 4-210) in order to request a media play list. In response, the web server 4-210 may determine whether the user of client 4-204 is authorized to receive the media play list associated with the request. The web server 4-210 may utilize a user database 4-302 in order to determine if a user is authorized to receive a media play list. If not, web server 4-210 does not provide client computer 4-204 the requested media play list. However, if client 4-204 is authorized, web server 4-210 transmits the media play list to the client device 4-204 (e.g., via Internet 4-202) along with an appended user key or user identification (ID).

It is understood that user database 4-302 may also include data for one or more media play lists that may be utilized to provide a media play list to client computer 4-204. Subsequently, the user of client device 4-204 may utilize the received media play list in combination with a media player application operating on client device 4-204 in order to transmit a delivery request (that automatically includes the user key or user ID) for one or more desirable pieces of media content to web server 4-210.

Upon reception of the media content delivery request, the web server 4-210 of Figure 4-3 checks the validity of the requesting media application along with the attached user key. The web server 4-210 may utilize user database 4-302 in order to check the validity of the user key (or user ID) and the requesting media application. If either or both are invalid, web server 4-210 redirects the unauthorized client computer 4-204 so that it may be handled in an appropriate manner in order to prevent abuse of the system. However, if both the requesting media application and user ID are valid, web server 4-210 issues to client device 4-204 a redirect command to the current address location of the desired media content along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe).

In response to reception of the redirect command, the media player application operating on client computer device 4-204 automatically issues a new request along with the time sensitive access key to content server 4-212 for delivery of the one or more desired pieces of media content. The content server 4-212 determines the validity of the time sensitive access key. If invalid, content server 4-212 redirects the unauthorized client device 4-204 so that it may be handled in an

appropriate manner in order to prevent access to the media content. However, if the time sensitive access key is valid, content server 4-212 retrieves the desired media content from a media content database 4-304 and then transmits it to client device 4-204 for use by its media player application. It is understood that the delivered media content may be stored by client computer 4-204 using hidden directories thereby preventing future unauthorized distribution of it.

Figures 4-4A and 4-4B are a flowchart 4-400 of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device (e.g., 4-204, 4-206 or 4-208). Flowchart 4-400 includes processes of the present invention which, in one embodiment, are carried out by a processor(s) and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions may reside, for example, in data storage features such as computer usable volatile memory 4-104, computer usable non-volatile memory 4-106 and/or computer usable mass data storage device 4-118. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 4-400, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 4-4A and 4-4B. Within the present embodiment, it should be appreciated that the steps of flowchart 4-400 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment enables delivery of "on-demand" high fidelity media content to computer devices via one or more communication networks (e.g., the Internet) while restricting unauthorized users from directly retrieving media content from a source database. Once the computer device receives the media, it may be stored using hidden directories so that it may not be easily shared with others. Within the present embodiment, there are different functionality that are implemented in order to protect and monitor the media content source. For example, the actual address location of the media database is hidden from content recipients while its address directory is periodically change making past addresses obsolete. Furthermore, an access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media content requests. By implementing these and other functionality, the present embodiment restricts access to and redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

It should be appreciated that flowchart 4-400 is described in conjunction with Figure 4-2 in order to more fully describe the operation of the present embodiment. At step 4-402 of Figure 4-4A, a client computer (e.g., 4-204) communicatively couples to a web server (e.g., 4-210) via one or more communication networks (e.g., Internet 4-202) and then proceeds to log in. The initial log in process of step 4-402 may be implemented in a wide variety of ways in accordance with the present embodiment. For example, the initial log in process may involve a registration process provided by web server 4-210 wherein different identifying information may be provided by a user of client computer 4-204 such as, but not limited to, his or her

name, address, phone number and credit card information. The present embodiment may then cause web server 4-210 to verify the truthfulness of the provided registration information. Furthermore, the registration process may include the user of client device 4-204 establishing with web server 4-210 a user selected identity and password.

However, if the user of client computer 4-204 has already participated in the registration process, the log-in process of step 4-402 may include web server 4-210 requesting that the user of client device 4-204 provide his or her previously established user identity and password. As such, the user of computer 4-204 causes it to transmit his or her user identity and password to web server 4-210. Upon receiving them, web server 4-210 determines their validity. If they are invalid, web server 4-210 refuses access to client computer 4-204 wherein flowchart 4-400 may be discontinued (not shown). However, if the provided user identity and password are valid, the present embodiment proceeds to step 4-404.

At step 4-404, once the registration process is completed, the present embodiment causes web server 4-210 to generate a unique user identification (ID) or user key associated with the user of client device 4-204 that participated in the initial log-in process of step 4-402. The unique user ID (or user key) is then stored by web server 4-210 in a manner that it is associated with that registered user. In this manner, the present embodiment is able to uniquely identify each authorized user of web server 4-210. It is appreciated that if a unique user ID has already been

generated for a user that logs in with web server 4-210 at step 4-402, the present embodiment may skip step 4-404 and proceed to step 4-406.

In step 4-406 of Figure 4-4A, the user of client computer 4-204 causes it to transmit to web server 4-210 (e.g., via Internet 4-202) a request for a play list of available media content. It is understood that the requested media play list may include all or a portion of the media content available from content server 4-212.

At step 4-408, in response to web server 4-210 receiving the play list request, the present embodiment causes web server 4-210 to transmit to client computer 4-204 a media content play list together with the unique user ID associated with the logged-in client. The user ID may be attached to the media content play list in a manner such that the user of computer 4-204 is unaware of it. It is appreciated that the media content of flowchart 4-400 may include, but is not limited to, high fidelity music, audio, video, graphics, multimedia, and the like. The media content play list of step 4-408 may be implemented in diverse ways. For example, the present embodiment may generate a media content play list by taking all of the media content titles available from the content server 4-212 and combining them into a single list. Alternatively, all of the media content titles (or different lists of titles) may be loaded from content server 4-212 and passed to a Common Gateway Interface (CGI) program operating on web server 4-210 that may be written in Perl (Practical Extraction and Report Language) Script, where the media titles (or different lists of titles) are concatenated into a single dimensioned array that may be provided to client device 4-204.

In step 4-410 of Figure 4-4A, the user of client computer 4-204 may utilize the received media content play list in conjunction with a media player application in order to cause client device 4-204 to transmit a request to web server 4-210 for delivery of particular media content, wherein the user ID automatically accompanies the request. The media delivery request of the present embodiment may be implemented in a wide variety of ways. For example, the media content play list provided to client device 4-204 by web server 4-210 may enable the user to create one or more customized media play lists by selecting desirable media content titles. It is understood that a customized media play list may establish the media content that eventually will be delivered to client computer 4-204 along with the sequential order in which it will be delivered. Furthermore, the user of client device 4-204 may create as many or as few customized play lists as desired and then store them with computer 4-204 (e.g., to its desktop) and/or within web server 4-210. It is noted that a customized media play list does not actually contain media content, but instead it includes one or more identifiers of specific pieces or portions of media content such as, but not limited to, a song, an audio clip, a video clip, a picture, a graphic picture, a multimedia clip, and the like.

The following Perl Script entitled "User Choice Program" is an exemplary implementation for providing a media play list to enable a user of a computer (e.g., 4-204) to generate a request for media delivery by creating one or more customized play lists.

```

#!/usr/bin/perl -w
#-----
#           User Choice Program
# This program reads in a mp3 play list source file, displays the file names,
# on a web page to allow for selection, and creates a new playlist file to download.
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
#-----
$VERSION = "5.0";

$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# ---- declare some variables used in the code program ----- #
# ---- Don't touch these --- #

my $dir_cnt = 0; #counter
my $qry1 = "";
my $qry2 = "";
my $sql = "";

# -----YOU CAN/SHOULD LOCALIZE THESE VARIABLES ----- #

my $rootdir = "mp3";
my $image_path = "http://www.themomi.org/museum/images";
my $image_url = "www.themomi.org/$rootdir";
my $song_root = "../htdocs/";
my $earthc = "themomi.earthc.net";

# ----- Start Code ----- #

$|=1; #flush
print "Content-type: text/html\n\n"; # send basic header
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();
use strict;

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
    "NOBODY", "ASK_FOR_JOE",
    { 'RaiseError' => 1 }) or die "can't connect to db";

my @play_list;
my $artist;
my $song_name;

print<<HTML_DONE;
<html>

```

```

<head>
<META NAME="Organization" CONTENT="www.themomi.org">
<META NAME="Author" CONTENT="Ted Fitzgerald">
<META NAME="Quote" CONTENT="My hovercraft is full of eels!">
<META NAME="Description" CONTENT="Music selection / creation software">
<SCRIPT LANGUAGE="JavaScript">
<!--
function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}
// -->
</SCRIPT>
<style><!--a { text-decoration:none }//--></style>
<style><!--a:hover { color:#3399cc; }--></style>

</head>
<body bgcolor="black">
<font size=2 color=white face = arial,san serif, helvetica >
<form method="post">

```

HTML_DONE

#----- Main page display starts here (selection portion ----- #

if first time page is displayed submit button hasn't been clicked and submit.blah is undefined.

we use this parameter to show initial selection or results portion of page

if(! defined(param("Submit.x"))){ #if first time page is displayed submit is undef

print<<HTML_DONE;

```

<center>
<a id="button">&nbsp;</a>
<a name="button">&nbsp;</a>
<input type="image" value="Submit" src="$image_path/oy.gif" border="0" alt="Create
Playlist" name="Submit">
<!-- <A HREF="http://www.themomi.org/perl/choose_hi.cgi"
ONMOUSEOVER="changeImages('oy',
'http://www.themomi.org/museum/images/oy.gif'); return true;"
ONMOUSEOUT="changeImages('oy',
'http://www.themomi.org/museum/images/oy_over.gif'); return true;">
<IMG NAME="oy" SRC="http://www.themomi.org/museum/images/oy.gif"
BORDER=0></A> -->
<br><br>
<input type=Reset value="Reset" name=Reset>
</center>

```

HTML_DONE

```

# ----- MAIN CODE STARTS HERE ----- #

my $sql = "select distinct playlist_name from m3u";
$sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);

    my $even = $dir_cnt % 2; # flip flop the output to make 2 columns odd = left even = right

    #make left right columns
    if(($even)) {
        print "<td align=center valign=top width=50%>";
    } else {
        print "<table align=center border=0 width=90% bordercolor=black name=outer>"; #start
        outer table
        print "<tr><td align=center valign=top width=50%>";
    } # end if

    print "<br><br><img src=http://$earhc/$image_url/$playlist_name/cover.jpg
    align=center><br><br>";

    print "<table border=0 align=center width=90% bordercolor=black
    name=internal_table>";

    print "<tr><td><input type=checkbox name=d$dir_cnt></td>";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >";
    print " --- Select All in Category ---</font>";
    print "</td></tr>";

    my $sql1 = "select pl_index,artist,song_name from m3u where playlist_name like
    \"\$playlist_name\"";
    # print "$sql1<br>";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();

    while (my $ref1 = $sth1->fetchrow_hashref()) {
        my $id = $ref1->{'pl_index'}; $artist = $ref1->{'artist'}; $song_name = $ref1-
        >{'song_name'};
        $artist =~ s/_/ /g;
        $song_name =~ s/_/ /g;
        print "<tr><td><input type=checkbox name=\"$id\"></td>\n";
        print "<td><font size=2 color=white face = arial,san serif, helvetica >$artist -
        $song_name<br></font></td></tr>\n";
    } #end while ref1

```

```

print "</table>"; # end inner table

if($seven) {
    print "</td></tr>";
    print "</table>"; #end outer table
    print "<center><br><font size=2 color=white face = arial,san serif, helvetica >";
    print "<a href=\"#button\">Return to top</a></font></center>";
}else{
    print "</td>";
} #end ending left right column if

$dir_cnt++; # increment counter used for left/right flip flop

} #end while ref -> display of play lists loop
#-----
print "<br><br>";

print "<input type=hidden name=size value=$dir_cnt>";

# $dbh->disconnect();

print " </form>";

} #end if (Submit)
#-- done displaying the selection list --

#-----#
# ----- DISPLAY RESULTS OF SELECTIONS ----- #
#
#-----#

if(defined(param("Submit.x")) || defined(param("Submit.x"))){
    my @play_list;
    my @list_copy;
    my @p_list;
    my $pl;
    my $first_pl;
    my $qry1;
    my $qry2;
    # $num_of_dirs = param("size");

#
=====
#my $sql = "select distinct playlist_name from m3u order by playlist_name asc";
my $sql = "select distinct playlist_name from m3u";
    $sql .= " order by playlist_name";
    my $sth = $dbh->prepare($sql);

```

```

$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);
} #end while "get all playlists" query
#
=====

# -----
# parse param list for "all in playlist" request = "d+some num" & individual songs "num"
# -----
my @song_list;
#my $pl_limit = 5;
#my $song_limit = 50;
#my $pl_count = 1;

foreach (param()) {
    push(@song_list,$_) if /^d+$/;    #just a song number
    # print "<br>$_" if /^d+$/;
    if (/^d+d+$/){
        #all in playlist d+number
        s/^d//;    #remove letter "d"

        push(@p_list,$play_list[$_]);    #store playlist number
        push(@list_copy,$play_list[$_]);

    } # end if "d+numb all in playlist "
} # end foreach "item in param() list"
# -----
#    Build "All in playlist" query string
# -----

if(@p_list){

    my $first_pl = shift @p_list;
    my @list_copy = $first_pl;
    $qry1 = "SELECT * FROM m3u WHERE playlist_name IN (\"$first_pl\"";

    foreach my $pl (@p_list){
        #    last if($pl_count = $pl_limit);
        #print "<br>=". @p_list; #print "<br>@p_list";

        $qry1 .= ", \"$pl\" ";

        push(@list_copy,$pl);
        #    $pl_count++;
    } #end foreach loop to build items in query string

    $qry1 .= ") order by playlist_name"; #    query ends here

```

```

# print "<br>$qry1";

} #end if anything in p_list array

# -----
#   Build individually selected songs query string
# -----
if(@song_list){ #anyone home in song lists? go for it!

    my @song_copy = @song_list;

    my $first = shift @song_list;
    $qry2 = "SELECT * FROM m3u WHERE pl_index IN ( $first" ;
    foreach my $blarg (@song_list){
        #print "<br>=". @song_list;
        #print "<br>@song_list";
        $qry2 .= ", $blarg ";
    } #
    $qry2 .= ")";
    # $qry2 .= " limit $song_limit"; #add limit to number of songs
    #print "<br>$qry2";
} #end if anything in song_list array

# -----
#   Now create m3u playlist file
#   Execute query statement(s) and write file
# -----

# print "<br>here's your check of array size greater than zero
defined($qry1),defined($qry2)<br>";
if($qry1 || $qry2){
    # ---- Create unique play list name using numbers from date parts ----- #
    my $theDate = localtime;
    $theDate =~ /(\d+):(\d+):(\d+)\b/i;
    my $play_list_name = "Momi_playlist_$2$3.m3u";
    # ----- #

    print "<center><font size=2 color=white face = arial,san serif, helvetica >";
    print "<b>Songs in your play list:</b></font></center>";
    print " <table align=center>";

    open(OUTFILE,">../htdocs/temp/$play_list_name");

    print OUTFILE '#EXTM3U'."\n"; #write file header
    #print '<br>#EXTM3U<br>'; #write file header

    if($qry1){
        my $list = "";

```

```

# now do the playlists
my $sth1 = $dbh->prepare($qry1);
$sth1->execute();
while (my $ref1 = $sth1->fetchrow_hashref()) {
    my $playlist_name = $ref1->{'playlist_name'};
    my $pl_header = $ref1->{'pl_header'};
    my $song_url = $ref1->{'song_url'};
    $song_url =~ s/8080/8081/g;
    print OUTFILE "$pl_header\n$song_url\n";
} # end while
#print "<br>my list_copy array = @list_copy</br><br>";
foreach my $list (@list_copy){
    $list =~ s/_/_/g;
    print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >All Songs
from <b>\'$list\'</b></font></td>
</tr>\n";
} # end foreach
}#end if $qry1 is defined

if($qry2){
    # now do the selected songs
    my $sth2 = $dbh->prepare($qry2);
    $sth2->execute();
    while (my $ref2 = $sth2->fetchrow_hashref()) {
        my $artist = $ref2->{'artist'};
        my $song_name = $ref2->{'song_name'};
        my $pl_header = $ref2->{'pl_header'};
        my $song_url = $ref2->{'song_url'};
        #print "$pl_header<br>$song_url<br>";
        $artist =~ s/_/_/g;
        $song_name =~ s/_/_/g;
        $song_url =~ s/8080/8081/g;
        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$ song_name</font></td></tr>";
        print OUTFILE "$pl_header\n$song_url\n";
    } # end while
}# end if $qry2 is defined

close OUTFILE;

print "</table>";
print "<center><br><br><br>";
print "<a href='\"http://www.themomi.org/temp/$play_list_name\"'>";
print "<img src='\"http://www.themomi.org/museum/newimages/create_now.gif\"'
border=0></a>";
print "</font>";
print "<br><br><br>";

# ----- include text for nubies ----- #

```

```

print "<br><br>";
print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the button above to launch
your playlist<br> with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>

<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the button above and
use "save target as" <br> to save your playlist file to your computer.</font>
</td></tr>

<tr><td colspan=3 align=center>
MORE

#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>

print "</table><br><br>";

}else{
  print "<br><b>Nothing Selected</b><br>";

  #foreach (param()) {
  #  print "$_<br>";
  #}#end foreach
  #print "The value of Param(\"Submit.x\") is ", param("Submit.x"), "<br>";
  #print "The value of defined Param(\"Submit.x\") is ", defined(param("Submit.x")),
  "<br>";

  $dbh->disconnect();

  }#end if defined queries
  }# end if (Submit.x) "display results of selection "

print "</body>";
print end_html();

```

#bye bye! Thanks for visiting!

exit(0);

1;

Alternatively, at step 4-410, the received media content play list may include a random media content delivery choice that the user of client device 4-204 may select to transmit a request to web server 4-210 for delivery of random media content, wherein the user ID automatically accompanies the request. An embodiment for implementing the delivery of media content in a random manner to client computer 4-204 is described in detail within reference to step 4-424.

At step 4-412 of Figure 4-4A, the present embodiment causes web server 4-210 to determine whether the requesting media application operating on client computer 4-204 is a valid media application. One of the main functions of a valid media application may be that it is a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If the web server 4-210 determines at step 4-412 that the requesting media application operating on the request client computer 4-204 is a valid media application, the present embodiment proceeds to step 4-416. However, if the web server 4-210 determines at step 4-412 that the requesting media application is not a valid application, the present embodiment proceeds to step 4-414.

In step 4-414, the present embodiment causes web server 4-210 to redirect client computer 4-204 so that it is prevented from accessing the source of the media content. It is understood that the redirection of step 4-414 may be performed in diverse ways. For example, the present embodiment may cause web server 4-210 to redirect client computer 4-204 to a software program that identifies it, logs it out of web server 4-210 and prevents any subsequent logging in of it for a defined amount of time (e.g., 10 minutes, an hour, a day, a month, a year, or any defined amount of time).

At step 4-416, the present embodiment causes the web server 4-210 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer 4-204 is valid. If the web server 4-210 determines that the user ID is not valid at step 4-416, the present embodiment proceeds to step 4-414. However, if the web server 4-210 determines that the user ID is valid at step 4-416, the present embodiment proceeds to step 4-418. It is noted that the order in which steps 4-412 and 4-416 are performed by the present embodiment may be switched from the order illustrated within Figure 4-4A. That is, the present embodiment may perform step 4-416 before performing step 4-412.

In step 4-418 of Figure 4-4A, the present embodiment causes web server 4-210 to transmit to client computer 4-204 a redirection command along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe) thereby enabling the client 4-204 to eventually receive the requested media content. The redirection command may include a time sensitive address of where the media

content is location within content server 4-212. The address is time sensitive because periodically the present embodiment of content server 4-212 renames some or all of the media address directories thereby making previous content source addresses obsolete. It is noted that the actual location of the media content may not change within content server 4-212, just the address of the content is changed. However, the locations of the media content may actually be changed along with the corresponding addresses. In any case, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 4-212. As such, if someone with inappropriate intentions is able to find out where the media content source is stored, the next time he or she tries to use that route (or address) it does not exist anymore thereby preventing further unauthorized access.

It is noted that within an embodiment of the present invention, the addresses (or routes) of content server 4-212 that are actively coupled to one or more client devices (e.g., 4-204 to 4-208) are maintained while future addresses (or routes) are being created for new client devices. It is appreciated that as client devices are uncoupled from the media content source of content server 4-212, that directory address (or link) may be immediately changed thereby restricting access to unauthorized client devices or applications.

Alternatively, the functionality of redirection to a media content source may be implemented by utilizing a server network where multiple servers are content providers (e.g., 4-212) or by routing a requesting client device (e.g., 4-204, 4-206 or

4-208) through multiple servers. Within another embodiment in accordance with the present invention, the delivery of media content from a central content provider (e.g., 4-212) may be routed through one or more intermediate servers before being received by the requesting client device (e.g., 4-204, 4-206 or 4-208).

The redirection functionality of step 4-418 is well suited to be implemented in a wide variety of ways. The following Perl Script entitled "Redirector Program" includes an exemplary implementation for the redirection functionality of step 4-418 along with other functionality of flowchart 4-400. Additionally, this Perl Script records the Internet Protocol (IP) addresses of the client computers (e.g., 4-204), the media content requested and its transfer size. In this manner, an implementation of flowchart 4-400 is able to accurately meter royalty payments, clock usage and transfers, and also keep a log of media content popularity.

```
#!/usr/bin/perl -w
```

```
#-----
#           Redirector Program
#-----
# This program obscures the proper address of the mp3 source file for due diligence
# purposes and records the requested mp3 file name, requestor ip number, file size
# and date of request to allow for accurate royalty payments & bandwidth monitoring.
# The resulting data is stored in a .csv file for ease of subsequent analysis.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
#----- #
```

```
my $version = 4.0;
```

```
#-----
$SIG{CHLD} = "IGNORE"; #kill zombies! Oooo scary scary!;
use IO::Socket::INET;
use DBI();
```

```

my $debug = 0; #debugging messages on/off

##### localize these variables #####
#
# server_port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra_low
# mp3_root_dir: the name of the root dir that contains the mp3 files
#   choices are: changing all the time
# real_dest_url: location of the content server
# send_away_url: location that browsers are sent to if the try to access mp3s.
#
#
#program name
my $level      = "hi";          # hi , mid, lo, x_lo;
my $short_name = $level . "_redir.cgi";
my $program_name = "http://www.themomi.org/redirs/$short_name";

# logs
my $apache_log = "/usr/local/apache/logs/access_$level.log";
my $error_log  = "/usr/local/apache/logs/error_$level.log";

#port
my $server_port = 8081;

#urls
my $real_dest_url = "www.mpb.tv";
my $send_away_url = "www.themomi.org";

#db connection
my $db_host = "AAA.BBB.CCC.DD";
my $db_user = "my user name";
my $db_pw   = "some password";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                      "$db_user", "$db_pw",
                      { 'RaiseError' => 1 }) or die "can't connect to db";

#rate control songs/min limit
my $limit = 10;

# Changeable access parameters read in from database
my $mp3_root_dir;
my $pass_key;

# change content directory names to prevent direct linking

my $sql_db = "Select dir from file_location ";
$sql_db .= "where band like \"\$level\"";
my $sth_db = $dbh->prepare($sql_db);
$sth_db->execute();
my $ref_db = $sth_db->fetchrow_hashref();

$mp3_root_dir = $ref_db->{'dir'};
$sth_db->finish();

if(!defined($mp3_root_dir)) {

```

```

print "oops!\n";
#-----
# $mp3_root_dir = "hi_band";
#   $mp3_root_dir = "Fatty";
# $mp3_root_dir = "ookii";
#-----
# $mp3_root_dir = "mid_band";
# $mp3_root_dir = "chu";
# $mp3_root_dir = "norman";
#-----
# $mp3_root_dir = "lo_band";
# $mp3_root_dir = "chisai";
# $mp3_root_dir = "Slim_Jim";
#-----
# $mp3_root_dir = "x_lo_band";
# $mp3_root_dir = "chibi";
# $mp3_root_dir = "Ally_McBeal";
#-----

} #end if

my $sql_key = "Select cache_key from cache_key ";
my $sth_key = $dbh->prepare($sql_key);
    $sth_key->execute();
my $ref_key = $sth_key->fetchrow_hashref();

    $pass_key = $ref_key->{'cache_key'};
    $sth_key->finish();

#####
my $msg = "debug mode off";
while ($_ = $ARGV[0]) {
    shift;
    #print "found $_ \n";
    last if /^-$/;
    if (/^-D/i || /^-X/i || /^-V/i) {
        $msg = "debug mode on";
        $debug = 1;
    }
}
}#end while

print "\nStarted program: $short_name Version: $version at Port: $server_port $msg \n";
print " Using $real_dest_url/$mp3_root_dir as music source dir\n";

$server = IO::Socket::INET->new(LocalPort=> $server_port,
                                Type    => SOCK_STREAM,
                                Reuse   => 1,
                                Listen  => 100 )
    or die "Couldn't be a tcp server on port $server_port: $!\n";

#####

# initialize variables used for player discrimination
my $ok = 1;    # on by default;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0;  # this is the winamp player
my $real = 0;    # this is the real player
my $ripper = 0;  # this is for stream ripper etc.

```

```

my $speeder = 0; # this is for speeders
my $what = "";
my $whole_request = "";
my $what_method = "";
my $what_song = "";
my $song_name = "song";
my $what_list = "list";
my $what_version = "";
my $first_line = "";
my $u_a_line = "";
my $ua = "unknown"; #generic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;

# ----- #
#   Now do main loop. fork off child when connection made
# ----- #

while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here

        my $client_info = getpeername($client);
        (my $port, my $iaddr) = unpack_sockaddr_in($client_info);
        my $remote_host_ip = inet_ntoa($iaddr);
        my $remote_hostname = gethostbyaddr($iaddr, AF_INET); #put in real name
        my $date = localtime;

        if(! defined($remote_hostname)) { $remote_hostname = "unknown";}
        if($remote_host_ip =~ /10.10.0.252/o || $remote_host_ip =~ /10.10.0.251/o){
            exit(1);
        }

        if($debug){
            print "\n ----- \n";
            print "$short_name: client ip = " . $remote_host_ip . " client_hostname = " . $remote_hostname . " \n";
        }

        #evaluate the first line
        $what = <$client>;
        #ignore network health check from x.x.x..104
        if(! defined($what) && $remote_host_ip !~ /209.10.35.104/){
            print "$short_name@$server_port encounterd a client who said nothing! $remote_host_ip , $date<--
\n "
;
            exit(1);
        }
        $first_line = $what;

        if($debug){ print "\n First line is \"", $what, "\" \n"; }
        if ($what !~ /^(GET|HEAD)\s+/i) { #valid requests must start with GET or HEAD
            if($remote_host_ip !~ /209.10.35.104/o){
                print "$short_name -> bad req HTTP/1.1 400, no GET HEAD, $remote_host_ip said:->$what<- \n";
            }
            print $client "HTTP/1.1 400, \"Bad Request\"";
            exit(1);
        }
    }
}

```

```

    }
    if ($what =~ /$mp3_root_dir/i) {
        print "$short_name -> bad req HTTP/1.1 400, mp3_dir in request, $remote_host_ip said:->$what<- \n";
        print $client "HTTP/1.1 400, \"Bad Request\"";
        exit(1);
    }
}

#banned ip list

if ($remote_host_ip =~ /132.239.12.77/
    || $remote_host_ip =~ /66.122.240.11/
    || $remote_host_ip =~ /63.17.233.236/) {
    #print " $short_name sending HTTP/1.1 400, baned $remote_host_ip \n said:->$what<- \n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

#now parse line, extract what we need or dump 'em
if ($what =~ /^([A-Z]+)\s+([\S]+)\s+HTTPV([\d\.]+)\s*/){
    $whole_request = $what;
    $what_method = $1;
    $what_song = $2;
    $what_version = $3;
}else{
    print " $short_name: sending bad client HTTP/1.1 400, Bad Request message\n";
    print " $short_name: bad req was: $what\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
    exit(1);
}

# print "method is > $what_method< song is >$what_song< version is >$what_version< \n";

if ($what_method !~ /(GET|HEAD)/i) {
    exit(1);
}

#now loop to get other info lines
while( ($request = <$client>) ne ("\"r\n" || "\"r\n\r\n") ){
    last if(! defined($request));
    $request =~ s/[r\n]//g;
    if($debug) {
        print "<$level: $counter >The client said: " . $request . "<\n";
    }
    #----- WEB BROWSER INFO ----- #
    if($request =~ /Mozilla/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $sua = $1;
    }
    if($request =~ /Opera/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $sua = $1;
    }
    if($request =~ /Omniweb/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $sua = $1;
    }
    if($request =~ /iCab/io) {
        $ok = 0; $mozzy = 1; $request =~ /\.+:\s*(.+)$/; $sua = $1;
    }
    #----- MP3 RIPPER INFO ----- #
    if($request =~ /StreamRipper/io) {
        $ok = 0; $ripper = 1; $request =~ /\.+:\s*(.+)$/; $sua = $1;
    }
}

```

```

#----- No get for Wget ----- #
if($request =~ /Wget/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- FlashGet fizzles ----- #
if($request =~ /FlashGet/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- GetRight is Wrong ----- #
if($request =~ /GetRight/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- Step on Spiders ----- #
if($request =~ /asterias/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- Web access = No access ----- #
if($request =~ /Web access/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}

#----- Monica gets the bone ----- #
if($request =~ /monica/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- man what the hell is this? ;- ) ----- #
if($request =~ /Media Jukebox WinInet Reader/io){
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- burn Nero Player ----- #
if($request =~ /NeroMediaPlayer/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- 1SMUSIC is deaf ----- #
if($request =~ /1SMUSIC/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}

#----- I know who Anonymizer is ----- #
if($request =~ /Anonymizer/io) {
    $ok = 0; $ripper = 1; $request =~ /\.*\s*(.+)$/; $ua = $1;
}
#----- MP3 CLIENT INFO ----- #
if($request =~ /Windows-Media-Player/io){
    $ok = 0; $nsplayer = 1; $request =~ /\.+\s*(.+)$/; $ua = $1;
}
if($request =~ /NSPlayer/io) {
    $ok = 0; $nsplayer = 1; $request =~ /\.+\s*(.+)$/; $ua = $1;
}
if($request =~ /RMA/io) {
    $ok = 0; $real = 1; $request =~ /\.+\s*(.+)$/; $ua = $1;
}
if($request =~ /Winamp/io) { $ok = 1; $request =~ /\.+\s*(.+)\s*$/; $ua = $1; }
if($request =~ /QuickTime/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }
if($request =~ /Xaudio/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }
if($request =~ /AppleApp/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }
if($request =~ /iTunes/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }
if($request =~ /Sonique/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }
if($request =~ /xmms/io) { $ok = 1; $request =~ /\.+\s*(.+)$/; $ua = $1; }

```

```

if($request =~ /UPlayer/io) { $ok = 1; $request =~ /\.+:\s*(.+)$/; $ua = $1;}

#### hold for error reportin ####
if($request =~ /User Agent/io) { $u_a_line = $request; }

last if ($counter > 20 ); #we don't want some client babbling on and on
$counter++;

}#end while

if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
}

##### Rate controls to prevent excessive access to content

$remote_host_ip =~ /\.+\.+/;
# my $last_octet_mod = $1;
# $last_octet_mod = $last_octet_mod % 1; #mod by number of tables we want
# my $time_offset = 1;

my $sql0 = "Select count(ip_address) as loser_count from loser ";
$sql0 .= "where ip_address = \"$remote_host_ip\" ";
$sql0 .= "AND user_agent = \"$ua\"";
my $sth0 = $dbh->prepare($sql0);
$sth0->execute();
my $ref0 = $sth0->fetchrow_hashref();
my $loser_count = $ref0->{'loser_count'};
$sth0->finish();

# #
if($loser_count <= 0 ){

my $sql = "SELECT COUNT(request_time) as number_songs_reqs ";
#$sql .= "MINUTE(MIN(request_time)) - MINUTE(now()) as duration ";
$sql .= "FROM rate_control_0 ";
$sql .= "WHERE ";
$sql .= "ip_address = \"$remote_host_ip\" ";
$sql .= "AND ";
$sql .= "user_agent = \"$ua\" ";
$sql .= "AND ";
$sql .= "request_time > now() - INTERVAL 60 SECOND";

print "\n$sql\n" if $debug;

my $sth = $dbh->prepare($sql);
$sth->execute();

my $ref = $sth->fetchrow_hashref();
my $song_count = $ref->{'number_songs_reqs'};
#my $duration = ($ref->{'duration'});
$sth->finish();

print "\n Song count is $song_count \n" if $debug;

if($song_count >= $limit){
    $speeder = 1;

```

```

    $ok = 0;
    print "\n Loser exceeded limit\n" if $debug ;

    my $sql1 = "insert into loser (ip_address,user_agent,loser_time) values
(\'$remote_host_ip\',\'$ua\',now()
)";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if song_count

}else{
    print "\n loser is in loser table\n" if $debug;
    $speeder = 1 ;
    $ok = 0;
}

if(!$speeder){
    my $sql1 = "INSERT INTO rate_control_0 ";
    $sql1 .= "(ip_address,user_agent,request_time) ";
    $sql1 .= "VALUES (\'$remote_host_ip\',\'$ua\',now())";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
}#end if

print $client "HTTP/$what_version 302 Found \r\n";
print " I told him HTTP/$what_version 302 Found \n" if $debug;
print $client "Date: $date \r\n";
print $client "Server: Apache/1.3.20 \r\n"; #lie to 'em to keep 'em guessing.
print $client "Keep-Alive: timeout=10, max=20\r\n";

#----- Good guys ----- #
if(($nsplayer || $real) && ! $speeder){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    print "found nsplayer or real player\n" if $debug;
}
if($ok && ! $speeder){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    # print "other player that doesn't need to be told what port to go to\n";
}
#----- Bad guys ----- #
if($mozzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if($ripper){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    # print "mozzy type web server being redirected to index page\n";
}
if( $speeder ){
    print "$short_name: Speeder @ $remote_host_ip sending bad client HTTP/1.1 400, Bad Request
message\n";
    print "$short_name: Speeder said $what\n";
    print $client "HTTP/1.1 400, \"Bad Request\"";
}

```

```
}
$ok = 1;
$nsplayer = 0;
$mozzy = 0;
$real = 0;

print $client "\n\n\n";
# print "\ndone\n";
# print "*****\n\n";

# $dbh->disconnect();
close $client;


$what_song =~ /\(.+)\V(.+)\.mp3/io;
$what_list = $1;
$song_name = $2;

#trash filter *****
if(! defined($what_list)){
    print "$short_name: no list\n"; $what_list = "none\n";
}
if(! defined($song_name)){
    print "$short_name: no song_name $remote_host_ip\n"; $song_name = "none";
    $bad = 1;
}
if(! defined($remote_hostname)){
    print "$short_name: no hostname\n"; $remote_hostname = "none";
    $bad = 1;
}
if(! defined($remote_host_ip)){
    print "$short_name: no host_ip\n"; $remote_host_ip = "none";
    $bad = 1;
}
if(! defined($remote_hostname)){
    print "$short_name: no hostname $remote_host_ip\n"; $remote_hostname = "none";
    $bad = 1;
}
if(! defined($ua)){
    print "$short_name: no ua\n"; $ua = "none";
    $bad = 1;
}

#match this: APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\")
#change date from: Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]

$date =~ /.+\s(.+)\s(.+)\s(.+)\s(.+)/io;
my $month_name = "$1";
my $day = "$2";
my $time = "$3";
my $year = "$4";

if ($day < 10){ $day = "0$day"; } #hack to add leading zero

$month_name =~ s/ //g;
my $apache_date = "[ $day/$1/$4:$3 -0500]";
my $apache_date = "[ $day/$month_name/$year:$time -0500]";
```

```

$whole_request =~ s/\r//;
$whole_request =~ s/\n//;

open (APACHE_LOG, ">>$apache_log") or warn "Waaaa! Can't open $apache_log";
$a_str = "$remote_host_ip - - $apache_date \"\$whole_request\" 302 2 \"\$program_name\" \"\$ua\"";
print APACHE_LOG "$a_str\n";
close APACHE_LOG;

if($bad || $ua eq "unknown" ){
    $first_line =~ s/[\r\n]//g;
    open (ERROR_LOG, ">>$error_log");
    my $e_str = "$remote_host_ip - - $apache_date \"\$first_line\" 400 2 \"\$short_name\" \"\$u_a_line\"";
    print ERROR_LOG "$e_str\n";
    close ERROR_LOG;
} #end if error

close STDOUT;
close STDERR;
close STDIN;

exit(0); # Child process return status and exits when done.
} # else 'tis the parent process, which goes back to accept()
}
$dbh->disconnect();
close ($server);
exit(0);

```

At step 4-420 of Figure 4-4B, upon receiving the redirection command, the present embodiment causes the media application operating on client computer 4-204 to automatically transmit to content server 4-212 a new media delivery request (that automatically includes the time sensitive access key) that includes the address of the desired media content.

In step 4-422, the present embodiment causes content server 4-212 to determine whether the time sensitive access key associated with the new media delivery request is valid. If the content server 4-212 determines that the time sensitive access key is not valid at step 4-422, the present embodiment proceeds to step 4-414 of Figure 4-4A. However, if the content server 4-212 determines that the

time sensitive access key is valid at step 4-422, the present embodiment proceeds to step 4-424.

At step 4-424 of Figure 4-4B, the present embodiment causes content server 4-212 to transmit the requested high fidelity media content to client computer 4-204. It is noted that when content server 4-212 transmits the media content to client computer 4-204 it does not encrypt the data stream. Additionally, content server 4-212 transmits the media content in a burst load (rather than a fixed data rate) that transfers to client device 4-204 as fast as its network transfer rate allows. The present embodiment may cause content server 4-212 to adapt its download speed to whatever network transfer rate the client computer's is able to handle. For example, if client computer 4-204 is coupled to the Internet 4-202 via a T1 communications line, content server 4-212 transfers the media content to client computer 4-204 at transmission speeds allowed by the T1 communications line. As such, once the requested high fidelity media content is transmitted to client computer 4-204, content server 4-212 is then able to transmit requested high fidelity media content to another client computer (e.g., 4-206 or 4-208). It should be appreciated that by performing step 4-424 in this manner, the delivery of high fidelity media content by the present embodiment becomes very efficient in terms of a statistical distribution over time and does not overly congest the utilized communication network(s).

It is appreciated that the delivery of the media content to client device 4-204 by content server 4-212 at step 4-424 may be implemented in a wide variety of ways. For example, while delivering the media content to client computer 4-204 at step 4-

424, a rate control restrictor functionality may be resident to content server 4-212 in order to monitor and limit "suspicious" media content retrieval. For instance, if a client device (e.g., 4-204) tries to retrieve media content from content server 4-212 faster than a predefined limit, the rate control restrictor uncouples client device 4-204 from its media content source (e.g., content server 4-212) and continues to restrict its access from the source for a predefined amount of time, as described herein. One of the reasons for implementing this rate control restrictor is to restrict access to those unauthorized applications that are designed to retrieve and/or copy media content from the source in an unauthorized manner. For example, a client device (e.g., 4-204) may be retrieving pieces of audio content at a rate determined to be much faster than is humanly possible to listen to in real time. As such, the rate control restrictor (which may be implemented with software and/or hardware) uncouples the client device from the media content source (e.g., content server 4-212) and restricts its access for a predefined amount of time.

Alternatively, an embodiment of the present invention may be implemented such that a user of a client device (e.g., 4-204) may establish a huge buffer of media content (e.g., audio clips, video clips, and the like) to be delivered by content server 4-212. However, the present embodiment at step 4-424 may just provide client computer 4-204 one piece of media content at a time as would be typically needed to utilize (e.g., listen to, watch, etc.) that piece of media content in real time.

Within another embodiment in accordance with the present invention, embedded keys and/or digital watermarks may be implemented within media content

stored by content server 4-212 which may be delivered during flowchart 4-400 of Figures 4-4A and 4-4B. By using embedded keys and/or digital watermarks within the media content, it is easier to determine if some unauthorized person has been retrieving and/or copying media content from a media content source of content server 4-212. The embedded keys and/or digital watermarks within media content may include, but are not limited to, information indicating where the media came from, the identity of the media requestor and/or the identity of the requestor client device (e.g., 4-204, 4-206 or 4-208). With this information, it may be determined by the present embodiment who or what client device has been unauthorized in retrieving and/or copying media content from one or more of the media sources (e.g., 4-212). As such, that person and/or client device (e.g., 4-204, 4-206 or 4-208) can be permanently restricted by the present embodiment from requesting and/or accessing media content.

It is understood that if the user of client computer 4-204 caused it at step 4-410 to transmit to web server 4-210 a request for delivery of random media content, this request information may be communicated to content server 4-212 in order for it to fulfill the request at step 4-424. The delivery of media content in a random manner at step 4-424 may be implemented in diverse ways in accordance with the present embodiment. For example, a random media play list may be generated by first taking all of the titles of the media content stored by content server 4-212 and concatenated them into a single dimensioned array. Next, a random number may be generated. It is appreciated that a random number may be generated in a wide

variety of ways in accordance with the present embodiment. For instance, a random number generator may incorporate as its product the combination of:

1. A number produced by a random function. It is noted that the random number may not truly be a random number, but a random sequence that follows a "seeded" number. This seeded number is a starting point for the further permutations of random selection, but would have been predictable with a fixed seed value.

2. The Julian date maintained within content server 4-212 that includes the hours, minutes, seconds, day, date, and four-digit year.

Using this combination, the following code may then be executed:

```
$theDate = 'date'; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "MoMI_random_playlist_$2$3.m3u";

$random_seed = "$3$2$3"; #yse time parts to make random
srand $random_seed; #seed the random num gen.
```

It is appreciated that the time/date provides a basis for seeding the random number generator that is repeatable once per millennium. Now that a random number has been selected, the actual process of applying this to the array of media content titles follows.

Exemplary criteria for selecting a media content title for inclusion in the random play lists may be, but is not limited to, three-fold:

1. There cannot be more than 50 media content titles total in the play list;

2. The selection should come from no more than 20 play lists stored by content server 4-212; and
3. The selection should not have been previously selected.

Assuming that the first two conditions have not been met, the resulting random number is used as the index into the media content title array and selects the song title that resides in that location in the array. If that title has been previously selected, a new random number is requested, and the process continues until all 3 conditions are fulfilled.

The resulting list of media titles are then formatted into a useable play list (or what may be referred to as a M3U file), and it is written to a memory device of content server 4-212 while a page is prepared for presenting this information to the user of client computer 4-204.

The following Perl Script entitled "Random Play List Program" is an exemplary implementation for enabling content server 4-212 to provide a random play list of media content as described herein.

```
#!/usr/bin/perl -w
#-----
#
#          RANDOM PLAY LIST PROGRAM
#
# This program reads in a mp3 play list source file in many directories ,
# displays the random choice of file names ,
# and creates a new file to download.
# Programmer: Ted Fitzgerald
# Created for "themomi.org"
#-----
```

```

my $version;
$version = 5.0;
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';

# declare variables

my ($i,$j); #just counters
my ($play_list_name); #variable to create random file name for download
my ($show_many_songs); # = total number of songs in all the dirs.
my (@thename); #array to contain the name of the randomly selected song
my (@thepath); #array to contain the URL path of the randomly selected song
my ($showname); #used to find the portion of name we want to display.
my ($song_url);
my (@theheader);
my (@artist);
my ($artist);
my ($theDate);
# ----- user ajustable paramerters -----
my $max_num_songs = 34; # set limit of how many songs to display
my $maxdirs = 20; #set limit on number of dirs to look in
my $mp3dir = "../htdocs/mp3"; #path from cgi script to mp3 directory
my $center_col_size = ($max_num_songs / 2 + 1);

$|=1; #flush the buffer
use strict;
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();

my $db_host = "AAA.BBB.CCC.DDD";

my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
    "PIZZA_MAN", "HE_DELIVERS",
    { 'RaiseError' => 1 }) or die "can't connect to db";

my $q = new CGI;

print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>Random Play List Creation</title>";
print "<style><!--a{ text-decoration:none }/--></style>";
print "<style><!--a:hover{ color:#3399cc; }--></style>";

print <<HTML_DONE;
<SCRIPT LANGUAGE="JavaScript">
<!--

function changeImages() {

```

```

    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}

// -->
</SCRIPT>
<head>
<body bgcolor="black" topmargin="0" leftmargin="0" marginheight="0" marginwidth="2"
text="white" link="#ffcc66" vlink="#ffcc
66" alink="#ffcc66">

<font size=2 color=white face = arial,san serif, helvetica >
HTML_DONE

$theDate = localtime; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "Momi_random_playlist_$3$2$3.m3u";
#-----#
# do all file io before displaying the names in list

my $sql = "select pl_header,song_name,song_url,artist from m3u order by rand() limit
$max_num_songs";

my $sth = $dbh->prepare($sql);
    $sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
    my $pl_header = $ref->{'pl_header'};
    my $song_name = $ref->{'song_name'};
    my $song_url = $ref->{'song_url'};
    my $artist = $ref->{'artist'};
    push (@thename,$song_name);
    push (@thepath,$song_url);
    push (@theheader,$pl_header);
    push (@artist,$artist);
}

open(OUTFILE,">../htdocs/temp/$play_list_name");
#write out file header
print OUTFILE "#EXTM3U\n";
for($i=0;$i<$max_num_songs;$i++){

    print OUTFILE "$theheader[$i]\n";
    $song_url = "$thepath[$i]";
    $song_url =~ s/8080/8081/io;
    print OUTFILE "$song_url\n";

```

```

}#end for
close OUTFILE;

#-----#
#now show 'em what they have in the list

print<<MORE;

<center>
MORE
print "<br>";
print "</center>";
print "<table align=\"center\" border=0 width = \"95%\">";
print "<tr width=45%><td><td rowspan=$center_col_size align=center valign=top>";

print <<HERE;
<A HREF="http://www.themomi.org/temp/$play_list_name"
    ONMOUSEOVER="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert_over.gif');
return true;"
    ONMOUSEOUT="changeImages('create_now_vert_01',
'http://www.themomi.org/museum/newimages/create_now_vert.gif'); return
true;">
    <IMG NAME="create_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/create_now_vert.gif"
BORDER=0></A>
HERE

print "</td>";
print "<td width=45%></td></tr>";
for($i=0;$i<$max_num_songs;$i++){
    my $flip_flop = ($i % 2 );

    $showname = $thename[$i];
    # $showname =~ /EXTINF:\d+,(+)/i;
    $showname =~ s/_/ /g;
    $artist = $artist[$i];
    $artist =~ s/_/ /g;

    if($flip_flop){
        print "<td width = 45% align=\"center\"><font size=2 color=white face = arial,san serif,
helvetica >$artist - $showname
</font></td></tr>";
    }else{
        print "<tr><td width= 45% align=\"center\"><font size=2 color=white face = arial,san
serif, helvetica >$artist - $showna
me </font></td>";
    }
}

```

```

}
print "</table><br>";

#-----#

# print "<center>";
# print "<br><a href=\"http://www.themomi.org/temp/$play_list_name\">";
# print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
# print "</center>";
print "<table align=center border=0 width=90%>";
print<<MORE;

<tr><td width=40% align=center>&nbsp;</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold" size="4">
MORE

print "<A HREF=\"http://www.themomi.org/perl/random_hi.cgi\"";

print<<MORE;
    ONMOUSEOVER="changeImages('rerandom_now_vert_01',
http://www.themomi.org/museum/newimages/re-randomize_sm_over.gif);
    return true;"
    ONMOUSEOUT="changeImages('rerandom_now_vert_01',
http://www.themomi.org/museum/newimages/re-randomize_sm.gif);
    return true;">
    <IMG NAME="rerandom_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/re-randomize_sm.gif"
BORDER=0></A>
</font></td>
<td>&nbsp;</td>
</tr>
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the center button above to launch
your <br>playlist with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold"
size="4">-- OR --</font></td>
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc"
size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the center button and
use "save target as" <br> to save your playlist file to your computer.</font>
</td></tr>
<tr><td colspan=3 align=center>
MORE

#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>

```

```

#Note: the music itself will not be saved locally <br> You will only be downloading one
small .m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
print "</table><br><br>";
print "<center>";

print $q->end_html;

1;

```

In step 4-426 of Figure 4-4B, upon receiving the requested high fidelity media content from the content server 4-212, the present embodiment causes client computer 4-204 to store it. The present embodiment may cause client device 4-204 to store the received media content in a manner such that it is not easy for the user of client computer 4-204 to redistribute the media content in an unauthorized manner. For example, the present embodiment may cause the high fidelity media content to be stored by a memory device of client device 4-204 utilizing one or more hidden directories where it may be cached for a limited amount of time. As part of restricting access to the media content, the present embodiment at step 4-412 may reject any media player application if it allows accessibility to one or more hidden directories. It is noted that if the user of client computer 4-204 turns it off or quits out of the media player application, the stored media content usually is typically deleted from the memory of client 4-204.

Alternatively, at step 4-426 the present embodiment may cause client device 4-204 to store the received media content in a non-temporary manner within the media application operating on it or within one of its Internet browser applications (e.g., Microsoft Internet Explorer™, Netscape Communicator™, and the like) such

that the media content may be utilized in a repetitive manner. Additionally, the received media content may be stored in a manner such that it is not easy for the user of client computer 4-204 to redistribute it in an unauthorized manner. For example, the stored media content may be implemented in a way such that it may be utilized by client device 4-204, but it may be restricted from being download to another client device (e.g., 4-206 or 4-208). It is noted that by storing the media content within client device 4-204 in this manner, content server 4-212 is relieved from redelivering the same media content to client device 4-204 each time its user desires to utilize (e.g., listen to, watch, etc.) it. Therefore, by not having to redeliver media content to client devices (e.g., 4-204 to 4-208), it is appreciated that flowchart 4-400 may be even more efficient in delivering media content to client devices.

At step 4-428, the user of client computer 4-204 may cause the media application operating on it to access and utilize the delivered high fidelity media content thereby enabling its user or users to experience (e.g., listen to, view, etc.) the media content. It is noted that a specialized or proprietary media player is not needed in order to utilize the received media content. Instead, an industry standard media player may be utilized by client computer 4-204 to utilize the received media content. Some of the exemplary industry standard media players that are typically available at no cost and are supported by longstanding organizations may include, but are limited to, Windows™ Media Player™ for personal computers (PCs), iTunes™ Player or Quicktime™ for Apple computers, and XMMS Player for computers utilizing the Linux operating system.

Figure 4-5 is a diagram of an exemplary global media content delivery system 4-500 in accordance with an embodiment of the present invention. System 4-500 may be utilized to globally deliver media content (e.g., audio, video, graphics, multimedia, etc.) to client devices (e.g., 4-204, 4-206 and/or 4-208 of Figure 4-2) in conjunction with any manner similar to that described herein. System 4-500 includes a global delivery network 4-502 that may include multiple content servers (e.g., 4-504, 4-506, 4-508, 4-510, 4-512, 4-514 and 4-516) located throughout the world which may be referred to as "points of presence." Each of content servers 4-504 to 4-516 may store a portion, a substantial portion or the entirety of a media content library that may be provided to client devices via a network, such as, the Internet (e.g., 4-202) or a wide area network (WAN). As such, each of content servers 4-504 to 4-516 may provide media content to client devices in its respective vicinity of the world.

For example, a content server (e.g., 4-516) located in Tokyo, Japan is able to provide media content from its stored media content library to client devices (e.g., 4-204, 4-206 and 4-208) within the Asian part of the world while a content server (e.g., 4-512) located in New York City is able to provide media content from its stored media content library to client devices within the eastern United States and Canada. It is noted that each city name (e.g., Hamburg, San Jose, Amsterdam, New York City, London or Tokyo) associated with one of the content servers 4-506 to 4-516 represents where that particular content server is located. However, these city name are exemplary since content servers 4-504 to 4-516 may be located anywhere in the world and are not limited to the locations shown within system 4-500.

It should be understood that system 4-500 is described in conjunction with Figures 4-2, 4-3 and 4-4 in order to more fully describe the operation of the present embodiment. Specifically, after a client device (e.g., 4-204) interacts with a central web server (e.g., 4-210) as described herein, the web server 4-210 may then redirect client device 4-204 to receive the desired media content from a content server (e.g., 4-504, 4-506, 4-508, 4-510, 4-512, 4-514 or 4-516) based on one or more different criteria.

For example, client device 4-204 may be located in Charlotte, North Carolina and its user causes it to log-in with a central web server 4-210 (which may be located anywhere in the world). It is understood that steps 4-402 to 4-416 of Figures 4-4A may then be performed as described herein such that the present embodiment proceeds to step 4-418 of Figure 4-4B. At step 4-418, the present embodiment may then determine which content server (e.g., 4-504 to 4-516) may subsequently provide the desired media content to client computer 4-204. The present embodiment may use one or more different criteria in determining which content server to choose. For example, the present embodiment may base its determination on which content server is located the shortest distance from client device 4-204. This may be performed by utilizing stored registration information (e.g., address) provided by the user of client computer 4-204. Alternatively, the present embodiment may base its determination on which content server provides media content to the part of the world that client device 4-204 is located within. However, if each of the content servers (e.g., 4-504 to 4-516) do not store the same media

content, the present embodiment may base its determination on which content server can actually provide the media content desired by client computer 4-204. It is noted that these are exemplary determination criteria and the present embodiment is not limited to such implementation.

Once the present embodiment determines which content server (e.g., 4-512) is to provide media content to client computer 4-204 at step 4-418, the present embodiment then causes web server 4-210 to transmit to client computer 4-204 a redirection command to content server 4-512 along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe) thereby enabling the client 4-204 to eventually receive the requested media content. Within system 4-500, the redirection command may include a time sensitive address of where the media content is location within the content server 4-512. As such, the New York City content server 4-512 may subsequently provide the desired media content to client device 4-204. It is noted that steps 4-418 to 4-428 and 4-414 of Figure 4-4B may then be performed with content server 4-512 in a manner similar to content server 4-212 as described herein.

One of the advantages of utilizing multiple content servers (e.g., 4-504 to 4-516) to provide high fidelity media content to client computers (e.g., 4-204 to 4-208) throughout the world is that communication network systems of the Internet do not become overly congested. Additionally, global network 4-502 is able to deliver media content to a much larger amount of client computers (e.g., 4-204 to 4-208) in a more responsive manner.

Within Figure 4-5, it is understood that content servers 4-504 to 4-516 of the global delivery network 4-502 may be coupled in a wide variety of ways in accordance with the present embodiment. For example, content servers 4-504 to 4-516 may be coupled utilizing wired and/or wireless communication technologies. Additionally, it is noted that content servers 4-504 to 4-516 may be functionally coupled such that if one of them fails for some reason, another content server may take over and fulfill its functionality. Furthermore, one or more web servers similar to web server 4-210 may be coupled to the global delivery network 4-502 utilizing wired and/or wireless communication technologies.

Within system 4-500, content server 4-504 includes a web infrastructure that is a fully redundant system architecture. It is understood that each content server (e.g., 4-506 to 516) of the global delivery network 4-502 may be implemented to include a web infrastructure in a manner similar to that implemented for content server 4-504.

Specifically, the web infrastructure of content server 4-504 includes firewalls 4-518 and 4-520 which are each coupled to global network 4-502. The firewalls 4-518 and 4-520 may be coupled to global network 4-502 in diverse ways in accordance with the present embodiment. For example, firewalls 4-518 and 4-520 may be coupled to global network 4-502 utilizing wired and/or wireless communication technologies. More specifically, firewalls 4-518 and 4-520 may each be coupled to global network 4-502 via a 10/100 Ethernet handoff. However,

system 4-500 is not in anyway limited to this particular implementation. It is understood that firewalls 4-518 and 4-520 are implemented in order to prevent malicious users from accessing in an unauthorized manner any part of the web infrastructure of content server 4-504, especially its stored media content. Firewalls 4-518 and 4-520 are coupled together. Furthermore, firewall 4-518 includes a device 4-536 and a DB server 4-540 while firewall 4-520 includes a device 4-538 and a DB server 4-542. Additionally, device 4-536 is coupled to firewall 4-518, DB servers 4-540 and 4-542. Moreover, device 4-538 is coupled to firewall 4-520, DB servers 4-542 and 4-540.

Within Figure 4-5, firewall 4-518 is coupled to a director device 4-522 which is coupled to an internal web application server 4-526, a hub server 4-530 and an internal web application server 4-528. The firewall 4-520 is coupled to a director device 4-524 which is coupled to internal web application servers 4-528 and 4-526, a staging server 4-534 and hub server 4-530. The hub server 4-530 may be implemented in diverse ways in accordance with the present embodiment. For example, hub server 4-530 may be specifically implemented as a Linux hub server. The hub server 4-530 is coupled to a storage device 4-532 capable of storing media content. Storage device 4-532 may be implemented in a wide variety of ways in accordance with the present embodiment. For example, storage device 4-532 may be implemented as a RAID (redundant array of independent disks) appliance.

It is understood that content servers 4-504 to 4-516 may each be implemented in any manner similar to content server 4-212 described herein.

Furthermore, content servers 4-504 to 4-516 of the present embodiment may each be implemented as one or more physical computing devices (e.g., computer 4-100 of Figure 4-1).

Accordingly, the present invention provides a method and system for delivering "on-demand" high fidelity music to computer systems via the Internet which does not involve proprietary audio players and/or encryption of the music. Additionally, the present invention provides a method and system that includes the above accomplishment and does not overly congest the communication networks of the Internet. Furthermore, the present invention provides a method and system that includes the above accomplishments and monitors the music (or media) delivered in order to compensate the owner of copyrighted music (or media) for it.

One embodiment of the present invention enables global delivery of "on-demand" high fidelity media content to client computers via a network, such as, the Internet or a wide area network (WAN) while restricting unauthorized users from directly retrieving media content from its sources. The present embodiment includes a global media content delivery network that may include multiple "points of presence" which may be located throughout the world. Each point of presence may store a portion or the entirety of a media content library that may be provided to client devices. Each one of the points of presence may provide media content to client devices in their respective vicinity of the world. Once a client receives media, it is stored using hidden directories to prevent easy redistribution with other devices.

An access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media requests.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 5: CONTROLLING INTERACTION OF DELIVERABLE ELECTRONIC
MEDIA

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not

necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop, handheld, and/or other portable electronic device. For purposes of the present application, the term "portable electronic device" is not intended to be limited solely to conventional handheld or portable computers. Instead, the term "portable electronic device" is also intended to include many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, and other personal digital devices.

Figure 5-1 is a block diagram illustrating an exemplary computer system 5-100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 5-100 can be nearly any type of computing system or electronic computing device including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 5-100 and which are executed by a processor(s) of computer system 5-100, in one embodiment. When executed, the instructions cause computer system 5-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 5-100 of Figure 5-1 comprises an address/data bus 5-110 for communicating information, one or more central processors 5-101 coupled to bus 5-110 for processing information and instructions. Central processor(s) 5-101 can

be a microprocessor or any alternative type of processor. Computer system 5-100 also includes a computer usable volatile memory 5-102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 5-110 for storing information and instructions for processor(s) 5-101. Computer system 5-100 further includes a computer usable non-volatile memory 5-103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc., coupled to bus 5-110 for storing static information and instructions for processor(s) 5-101. In one embodiment, non-volatile memory 5-103 can be removable.

System 5-100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 5-104 coupled to bus 5-110 for enabling computer 5-100 to interface with other electronic devices. Communication interface 5-104 can include wired and/or wireless communication functionality. For example, in one embodiment, communication interface 5-104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire (IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device 5-104. In such an instance, communication interface 5-104 may include a DSL modem.

Computer 5-100 of Figure 5-1 can also include one or more computer usable data storage device(s) 5-108 coupled to bus 5-110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 5-108 can be a magnetic storage device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 5-108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks) configuration. It is noted that data storage device 5-108 can be located internal and/or external of system 5-100 and communicatively coupled with system 5-100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 5-100. It is further noted that nearly any portable electronic device, e.g., device 5-100a, can also be communicatively coupled with system 5-100 via utilization of wired and/or wireless technology, thereby expanding the functionality of system 5-100.

System 5-100 can also include an optional display device 5-105 coupled to bus 5-110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 5-105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma display, a field emission display (FED) or any other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 5-100 of Figure 5-1 further includes an optional alphanumeric input device 5-106 coupled to bus 5-110 for communicating information and command selections to processor(s) 5-101, in one embodiment. Alphanumeric input device 5-106 is coupled to bus 5-110 and includes alphanumeric and function keys. Also included in computer 5-100 is an optional cursor control device 5-107 coupled to bus 5-110 for communicating user input information and command selections to processor(s) 5-101. Cursor control device 5-107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, a optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 5-106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 5-100 is configured with such functionality.

Figure 5-2 is a block diagram of an exemplary network 5-200 in which embodiments of the present invention may be implemented. In one example, network 5-200 enables one or more authorized client computer systems (e.g., 5-210, 5-220, and 5-230), each of which are coupled to Internet 5-201, to receive media content from a media content server, e.g., 5-251, via the Internet 5-201 while preventing unauthorized client computer systems from accessing media stored in a database of content server 5-251.

Network 5-200 includes a web server 5-250 and a content server 5-251 which are communicatively coupled to Internet 5-201. Further, web server 5-250 and content server 5-251 can be communicatively coupled without utilizing Internet 5-201, as shown. Web server 5-250, content server 5-251, and client computers 5-210, 5-220, and 5-230 can communicate with each other. It is noted that computers and servers of network 5-200 are well suited to be communicatively coupled in various implementations. For example, web server 5-250, content server 5-251, and client computer systems 5-210, 5-220, and 5-230 of network 5-200 can be communicatively coupled via wired communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 5-2, it is noted that web server 5-250, content server 5-251, and client computer systems 5-210, 5-220 and 5-230 of network 5-200 can, in one embodiment, be each implemented in a manner similar to computer system 5-100 of Figure 5-1. However, the server and computer systems in network 5-200 are not limited to such implementation. Additionally, web server 5-250 and content server 5-251 can perform various functionalities within network 5-200. It is also noted that, in one embodiment, web server 5-250 and content server 5-251 can both be disposed on a single or a plurality of physical computer systems, e.g., computer system 5-100 of Figure 5-1.

Further, it is noted that network 5-200 can operate with and deliver any type of media content, (e.g., audio, video, multimedia, graphics, information, data, software

programs, etc.) in any format. In one example, content server 5-251 can provide audio and video files to client computers 5-210 to 5-230 via Internet 5-201.

Figure 5-3 is a block diagram of an exemplary copyright compliance mechanism (CCM) 5-300, for controlling distribution of, access to, and/or copyright compliance of media files, in accordance with an embodiment of the present invention. In one embodiment, CCM 5-300 contains one or more software components and instructions for enabling compliance with DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry association of America) licensing agreements regarding media files.

There are currently two types of copyright licenses recognized by the DMCA for the protection of broadcasted copyrighted material. One of the broadcast copyright licenses is a compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one example, the period of time can be three hours.

The other type of the broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright

holder grants permission for a server, e.g., web server 5-250 and/or content server 5-251 of Figure 5-2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with “no save” license, meaning that the end user is unable to save or store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user to save that particular music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This is commonly referred to as an interactive with “save” license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

It is noted that the DMCA allows for the “perfect” reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording.

In one embodiment, CCM (copyright compliance mechanism) 5-300 can be stored in web server 5-250 and/or content server 5-251 of network 5-200 and which is configured to be installed into each client computer system, e.g., 5-210, 5-220 and 230, that is enabled to access the media files stored within content server 5-251

and/or web server 5-250. Alternatively, copyright compliance mechanism 5-300 can be, in another embodiment, externally disposed and communicatively coupled with a client computer system, e.g., system 5-210. In one embodiment, portions of components, entire components and/or combinations of components of CCM 5-300 can be readily updated, e.g., via Internet 5-201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or licensing agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications.

Referring to Figure 5-3, in one embodiment, CCM 5-300 is shown to include instructions 5-301 for enabling client computer system 5-210 to interact with web server 5-250 and content server 5-251 of network 5-200. Instructions 5-301 enable client computer system 5-210 to interact with servers, e.g., 5-250 and 5-251 in a network, e.g., 5-200.

The copyright compliance mechanism 5-300 also includes, in one embodiment, a user ID generator 5-302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 5-210. In one embodiment, the user ID and the cookie(s) are installed in computer system 5-210 prior to installation of the remaining components of the copyright compliance mechanism 5-300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 5-250 before the remaining components of a CCM 5-300 can be installed, within one embodiment of the present invention. Additionally, the user ID/user key can contain,

but is not limited to, the user's name, the user's address, the user's credit card number, verified email address, and an identity (username) and password selected by the user. Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 5-210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 5-210, e.g., a MAC (machine address code) address and/or an IP address, and other information specific to the user and the computer system operated by the user. It is noted that the information regarding the client computer system, e.g., 5-210, the user of system 5-210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 5-210, web server 5-250 can determine when a user of one computer system, e.g., 5-210, has given their username and password to another user using another computer system, e.g., 5-220. Because the username, password, and the user's computer system 5-210 are closely associated, web server 5-250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 5-250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 5-210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 5-250. Web server 5-250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 5-3, copyright compliance mechanism 5-300 further includes one or more coder/decoders (codec) 5-303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 5-300. In the present embodiment, coder/decoder 5-303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 5-251. In the present embodiment, coder/decoder 5-303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 5-300 rules are being enforced at any particular moment during media playback. It is noted that differing coder/decoders 5-303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 5-3, copyright compliance mechanism 5-300 also includes one or more agent programs 5-304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 5-210, 5-220, or 5-230, a server, e.g., web server 5-250 and/or content server 5-251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 5-304 can also be configured to maintain

system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 5-300 also includes one or more system hooks 5-305, in one embodiment of the present invention. A system hook 5-305 is, in one embodiment, a library that is installed in a computer system, e.g., 5-210, and intercepts system wide events. For example, a system hook 5-305, in conjunction with skins 5-306, can govern certain properties and/or functionalities of media player applications operating within the client computer system, e. g., 5-210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during playing of a media file, the

progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that system hook 5-305 compares the information for the media player application operating in client computer system, e.g., 5-210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 5-300 described herein.

The following C++ source code is exemplary implementation of the portion of a system hook 5-305 for performing media player application detection, in accordance with an embodiment of the present invention.

```
int
IsRecorderPresent(TCHAR *    szAppClass,
                  TCHAR *    szProdName)
{
    TCHAR    szWndText[_MAX_PATH]; /* buffer to receive title string for
window */
    HWND     hWnd;                /* handle to target window for operation */
    int      nRetVal;             /* return value for operation */

    /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
        == 0)
```

```

    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
            != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
    else
    {
        /* attempt to locate window with specified class */
        if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
            != (HWND) 0)
        {
            /* attempt to retrieve title string for window */
            if ( GetWindowText(hWnd,
                               szWndText,
                               _MAX_PATH)
                != 0)
            {
                /* attempt to locate product name within title string */
                if ( _tcsstr(szWndText, szProdName)
                    != (TCHAR *) 0)
                {
                    /* indicate application found */
                    nRetVal = 1;
                }
            }
        }
    }

    /* return to caller */
    return nRetVal;
}

```

It is further noted that system hook 5-305 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 5-210. For example, system hook 5-305, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 5-210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data

pathways that are commonly used for recording, such that the utilized data pathway is governed by a copyright compliance mechanism 5-300. This can be performed within a driver shim for a standard Window™ waveform output device, e.g., Windows™ Media Player. Client computer system 5-210 is configured such that the driver shim will appear as the default waveform audio device to client level application programs. Thus, requests for processing of waveform audio input and/or output will pass through the driver shim prior to being forwarded to the actual waveform audio driver. Such waveform input/output suppression can be triggered by other components of CCM 5-300, e.g., agent 5-304, to be active when a recording operation is initiated by a client computer system, e.g., 5-210, during the play back of media files which are subject to the DMCA. It is noted that alternative driver shims can be implemented for nearly any waveform output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices.

The following C++ source code is an exemplary implementation of the portion of a system hook 5-305 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```

DWORD
_stdcall
widMessage(UINT          uDevId,
            UINT          uMsg,
            DWORD         dwUser,
            DWORD         dwParam1,

```

```

        DWORD    dwParam2)
{
    BOOL        bSkip;        /* flag indicating operation to be skipped */
    HWND        hWndMon;      /* handle to main window for monitor */
    DWORD        dwRetVal;     /* return value for operation */

    /* initialize variables */
    bSkip = FALSE;
    dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;

    if (uMsg == WIDM_START)
    {
        /* attempt to locate window for monitor application */
        if ( (hWndMon = FindMonitorWindow())
            != (HWND) 0)
        {
            /* obtain setting for driver */
            bDrvEnabled = ( SendMessage(hWndMon,
                                      uiRegMsg,
                                      0,
                                      0)
                          == 0)
                        ? FALSE : TRUE;
        }

        if (bDrvEnabled == TRUE)
        {
            /* indicate error in operation */
            dwRetVal = MMSYSERR_NOMEM;

            /* indicate operation to be skipped */
            bSkip = TRUE;
        }
    }

    if (bSkip == FALSE)
    {
        /* invoke entry point for original driver */
        dwRetVal = CallWidMessage(uDevId, uMsg, dwUser, dwParam1,
dwParam2);
    }

    /* return to caller */
    return dwRetVal;
}

```

It is noted that when properly configured, system hook 5-305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 5-210 to 5-230. In one embodiment, system hook 5-305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a Microsoft Windows™ operating environment, and accordingly can be implemented in a variety of ways. However, it is also noted that system hook 5-305 can be readily adapted for implementation in alternative operating system, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 5-3, copyright compliance mechanism 5-300 also includes one or more skins 5-306, which can be designed to be installed in a client computer system, e.g., 5-210 to 5-230. In one embodiment, skins 5-306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 5-306 are customizable interfaces that, in one embodiment, are displayed on a display device (e.g., 5-105) of computer system 5-210 and provide functionalities for user interaction of delivered media content. Additionally, skins 5-306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

Furthermore, when system hook 5-305 is unable to govern a function of the media player application operable on a client computer system, e.g., 5-210, such

that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 5-306 can be implemented to provide compliance.

Differing skins 5-306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 5-306a may be configured for utilization with a media content file protected under a non-interactive agreement (DMCA), such that skin 5-306a may not include a pause function, a stop function, a selector function, and/or a save function, etc. Another skin, e.g., skin 5-306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with "no save" agreement (DMCA), such that skin 5-306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with "save" agreement, a save or a burn to CD function.

Still referring to Figure 5-3, it is further noted that in the present embodiment, each skin 5-306 can have a unique name and signature. In one embodiment, skin 5-306 can be implemented, in part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD 5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 5-306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD 5 hash, without knowledge of the MD 5 hash table, changing the name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance

mechanism 5-300 verifies skin 5-306, MD5 hash tables advantageously provide a deterrent against skin name changes and /or modifications made thereto.

In one embodiment, copyright compliance mechanism 5-300 also includes one or more custom media device driver(s) 5-307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 5-210, is configured to utilize a custom media device application, e.g., a custom audio device application, a custom video device application, etc., that is emulated by a custom media device driver 5-307. With reference to audio media, the emulation is performed in a waveform audio driver associated with a custom audio device. Driver 5-307 is configured to receive a media file being outputted by system 5-210 prior to the media file being sent to a media output device, e.g., a video card for video files or a sound card for audio files, etc. In one embodiment, client computer system 5-210 is configured with a custom media device driver 5-307 as the default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of a skin 5-306 or a custom web based player application, for forcing or requiring system 5-210 to have driver 5-307 as the default driver.

Therefore, when a media content file is received by system 5-210 from server 5-251, the media content file is playable, provided the media content file passes through the custom media device application, emulated by custom media device driver 5-307, prior to being outputted. However, if an alternative media player

application is selected, delivered media files from server 5-251 will not play on system 5-210.

Thus, secured media player applications would issue a media request to the driver for the custom media device which then performs necessary media input suppression, e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 5-307 to a Windows™ waveform audio driver operable on system 5-210, thus reducing instances of incompatibilities with existing media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 5-300, e.g., agents 5-304, system hooks 5-305, and skins 5-306, or a combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of ways including, but limited to, those found at developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 5-210 being configured with a custom media device, emulated by a custom media device driver 5-307, as the default device driver, those media player applications that require their particular device driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured music. Further advantageous is that an emulated custom media device provides no native support for those media player applications used as a recording mechanism, e.g., DirectSound capture, etc., that are able to bypass user-mode drivers for most media devices. Additionally, by virtue of the media content being sent through device driver 5-307, thus effectively disabling unauthorized saving/recording of media files, in one embodiment, media files that are delivered in a secured delivery system do not have to be encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-encrypted media files utilizing less storage space and network resources than encrypted media files, networks having limited resources can utilize the functionalities of driver 5-307 of CCM 5-300 to provide compliance with copyright restrictions and/or licensing agreements applicable with a media content file without having the processing overhead of encrypted media files.

Figure 5-4 is an illustration of an exemplary system 5-400 for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention. Specifically, system 5-400 illustrates web server 5-250, content server 5-251, or a combination of web server 5-250 and content server 5-251 installing a copyright compliance mechanism (e.g., 5-300) in a client's computer system (e.g., 5-

210) for controlling media file distribution and controlling user access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 5-210 can communicatively couple with a network (e.g., 5-200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 5-250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 5-250 can initiate a registration process with the requesting client 5-210. Client registration can be accomplished in a variety of ways. For example, web server 5-250 may deliver to a client 5-210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 5-250 including, but not limited to, user's name, address, phone number, credit card number, verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 5-4, web server 5-250 can, in one embodiment, detect information related to the client's computer system, e.g., 5-210, and store that information in a user/media database 5-450. For example, web server 5-250 can detect a unique identifier of client computer system 5-210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network interface card) of client computer system 5-210 or the MAC address of the network interface adapter integrated on the motherboard of system 5-210. It is understood

that a NIC enables a client computer system 5-210 to access web server 5-250 via Internet 5-201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 5-250 can, in one embodiment, detect and store (also in database 5-450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 5-210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 5-450) within web server 5-250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 5-210, all or part of this information can be installed in client computer system 5-210. In one embodiment, client computer system 5-210 information can be in the form of a cookie. Web server 5-250 then verifies that the user and client computer system 5-210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 5-250 installs a copyright compliance mechanism (e.g., 5-300) into the client's computer system, e.g., 5-210, in one embodiment of the present invention. It is noted that web server 5-250 may not initiate installation of CCM 5-300 until the user ID, password, and client computer system 5-210 information is verified. A variety of common techniques can be employed to install CCM 5-300. For example, copyright compliance mechanism 5-300 can be installed in a hidden directory within client computer system 5-210, thereby preventing unauthorized

access to it. In one embodiment of the present invention, it is noted that unless CCM 5-300 is installed in client computer system 5-210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 5-250 and/or content server 5-251,

Referring still to Figure 5-4, upon completion of client registration and installation of CCM 5-300, client computer system 5-210 can then request a media play list or a plurality of play lists, etc. In response, web server 5-250 determines whether the user of client computer system 5-210 is authorized to receive the media play list associated with the request. In one embodiment, web server 5-250 can request the username and password. Alternatively, web server 5-250 can utilize user database 5-450 to verify that computer 5-210 is authorized to receive a media play list. If client computer 5-210 is not authorized, web server 5-250 can initiate client registration, as described herein. Additionally, web server 5-250 can disconnect computer 5-210 or redirect it to an alternative web site. Regardless, if the user and client computer system 5-210 are not authorized, web server 5-250 will not provide the requested play list to client computer system 5-210.

However, if client computer system 5-210 is authorized, web server 5-210 can check copyright compliance mechanism 5-300 within data base 5-450 to determine if it, or any of the components therein, have been updated since the last time client computer system 5-210 logged in to web server 5-250. If a component of CCM 5-300 has been updated, web server 5-250 can install the updated component and/or a more current version of CCM 5-300 into client computer system 5-210, e.g., via

Internet 5-201. If CCM 5-300 has not been updated, web server 5-250 can then deliver the requested media play list to system 5-210 via Internet 5-201 along with an appended user key or user identification (ID). It is noted that user database 5-450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 5-210. Subsequently, the user of client computer system 5-210 can utilize the received media play list in combination with the media player application operating on system 5-210 to transmit a delivery request for one or more desired pieces of media content from web server 5-250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 5-4, upon receiving the media content delivery request, web server 5-250 can then check the validity of the requesting media application and the attached user key. In one embodiment, web server 5-250 can utilize user database 5-450 to check their validity. If either or both are invalid, web server 5-250, in one embodiment, can redirect unauthorized client computer system 5-210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 5-300 verifies that skins 5-306 are installed in client computer system 5-210. Additionally, CCM 5-300 further verifies that system hook(s) 5-305 have been run or are running to govern certain functions of those media player applications operable within client computer system 5-210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 5-300 further diverts and/or redirects certain pathways that are commonly used for recording. Once CCM 5-300 has performed the above described functions, web server 5-250 then, in one embodiment, issues to the client

computer 5-210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 5-210 receiving the redirect command from web server 5-250, the media player application operating on client computer system 5-210 automatically transmits a new request and the time sensitive access key to content server 5-251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 5-251. If invalid, unauthorized client computer 5-210 is redirected by content server 5-250 to protect against abuse of the system and unauthorized access to content server 5-251. If the time sensitive access key is valid, content server 5-251 retrieves the desired media content from content database 5-451 and delivers it to client computer system 5-210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 5-210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 5-210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 5-4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g.,

title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 5-300 that identifies the media file as originating from a content server 5-251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 5-300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or the entire CCM 5-300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to modification of copyright compliance mechanism 5-300, modified portions of, or the entire updated CCM 5-300 can easily be installed in client computer system 5-210 in a variety of ways. For example, the updated CCM 5-300 can be installed during client interaction with web server 5-250, during user log-in, and/or while client computer system 5-210 is receiving the keyed play list.

Referring still to Figure 5-4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 5-251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be

stored in a database 5-450, content database 5-451 or in some combination of databases 5-450 and 5-451. It is further noted that the messages being passed back and forth between client computer system 5-210 and web server 5-250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting with adds, public domain encryption programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be similarly encrypted. Furthermore, in another embodiment, the media files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 5-307.

Subsequent to media file decryption, the media file may be passed through CCM 5-300, e.g., a coder/decoder 5-303, to a media player application operating on client computer system 5-210 which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 5-306 of Figure 5-3. A skin 5-306 may be necessary when CCM 5-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client

computer system 5-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 5-303 will repeatedly ensure that CCM 5-300 rules are being enforced at any particular moment during media playback, shown as step 5-550 of Figure 5-5C.

As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 5-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 5-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These presentation stoppages are collectively shown as step 5-551 of Figure 5-5C.

If the rules, in accordance with CCM 5-300, are enforced, the codec/decoder 5-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 5-210. The newly retrieved portion of the media file is then presented by the client's media player application. While the newly retrieved portion is presented, CCM 5-300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 5-300 can detect undesired activities and enforces those rules as defined by CCM 5-300.

Figures 5-5A, 5-5B, and 5-5C, are a flowchart 5-500 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 5-500 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 5-104 and/or computer usable non-volatile memory 5-103 of Figure 5-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 5-500, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps

recited in Figures 5-5A, 5-5B, and 5-5C. Within the present embodiment, it should be appreciated that the steps of flowchart 5-500 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for controlling interaction of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 5-300, can be installed in the client computer system 5-210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 5-500 is described in conjunction with Figures 5-2, 5-3, and 5-4, in order to more fully describe the operation of the present embodiment. In step 5-502 of Figure 5-5A, a user of a computer system, e.g., 5-210, causes the computer to communicatively couple to a web server, e.g., 5-250, via one or more communication networks, e.g., Internet 5-201, and proceeds to attempt to log in. It is understood that the log in process of step 5-502 can be accomplished in a variety of ways in accordance with the present invention.

In step 5-504 of Figure 5-5A, web server 5-250 accesses a user database, e.g., 5-450, to determine whether the user and the computer system 5-210 logging in are registered with it. If the user and computer system 5-210 are registered with web server 5-250, the present embodiment proceeds to step 5-514. However, if the user and computer system 5-210 are logging in for the first time, web server 5-250 can initiate a user and computer system 5-210 registration process at step 5-506.

In step 5-506, registration of the user and computer system 5-210 is initiated. The user and computer system registration process can involve the user of computer system 5-210 providing personal information including, but not limited to, their name, address, phone number, credit card number, and the like. Web server 5-250 can verify the accuracy of the information provided. Web server 5-250 can also acquire information regarding the user's computer system 5-210 including, but not limited to, identification of media players disposed and operable on system 5-210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address.

Additionally, web server 5-250 can further request that the user of computer system 5-210 to select a username and password.

In step 5-508 of Figure 5-5A, subsequent to the completion of the registration process, web server 5-250 generates a unique user identification (ID) or user key associated with the user of client computer system 5-210. The unique user ID, or user key, is then stored by web server 5-250 in a manner that is associated with that registered user. Furthermore, one or more cookies containing that information specific to that user and the user's computer system 5-210, is installed in a non-volatile memory device, e.g., 5-103 and/or data storage device 5-108 of computer system 5-210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 5-210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user key, has been previously generated for the user and computer 5-210 that initially logged-in at step 5-502, the present embodiment proceeds to step 5-514

In step 5-510, web server 5-250 verifies that the user ID and the cookie(s) are properly installed in computer system 5-210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 5-250 can re-initiate the registration process at step 5-506. Alternatively, web server 5-250 can decouple computer system 5-210 from the network, thereby requiring a re-log in

by the user of computer 5-210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 5-512.

In step 5-512 of Figure 5-5A, web server 5-250 can install a version of a copyright compliance mechanism 5-300 into one or more non-volatile memory devices of computer system 5-210. Installing CCM 5-300 into user's computer system 5-210 facilitates client side compliance with licensing agreements and copyright restrictions applicable to specific delivered copyrighted media content. At step 5-512, the components of CCM 5-300, such as instructions 5-301, coder/decoder (codec) 5-303, agent programs 5-304, system hooks 5-305, skins 5-306, and custom media device drivers 5-307, are installed in computer system 5-210. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 5-300 into computer system 5-210. However, step 5-512 is well suited to install CCM 5-300 on computer system 5-210 in a wide variety of ways in accordance with the present embodiment.

In step 5-514, web server 5-250 can request the previously established username and password of the user of client computer system 5-210. Accordingly, the user of client computer system 5-210 causes it to transmit to web server 5-250 the previously established username and password. Upon the receipt thereof, web server 5-250 may access a user database, e.g., 5-450, to determine their validity. If the username and password are invalid, web server 5-250 refuses access wherein flowchart 5-500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 5-516.

In step 5-516 of Figure 5-5A, web server 5-250 can access media file database 5-450 to determine if copyright compliance mechanism 5-300 has been updated to reflect changes made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 5-300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 5-300 can provide compliance with current copyright restrictions.

Continuing with step 5-516, if web server 5-250 determines that CCM 5-300, or components thereof, of computer 5-210 has been updated, web server 5-250 initiates installation of the newer components and/or the most current version of CCM 5-300 into computer system 5-210, shown as step 5-518. If web server 5-250 determines that the current version of CCM 5-300 installed on system 5-210 does not have to be updated, the present embodiment proceeds to step 5-520 of Figure 5-5B.

In step 5-520 of Figure 5-5B, the user of client computer system 5-210 causes it to transmit to web server 5-250, e.g., via Internet 5-201, a request for a play list of

available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g., 5-251.

In step 5-522, in response to web server 5-250 receiving the play list request, web server 5-250 transmits to client computer system 5-210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media content play list in a manner invisible to the user. It is noted that the media content in content server 5-251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, and the like. The media content play list of step 5-520 can be implemented in diverse ways. In one example, web server 5-250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media content titles, or different lists of titles, can be loaded from content server 5-251 and passed to a CGI (common gateway interface) program operating on web server 5-250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 5-210. It is understood that the CGI can be written in nearly any software computing language.

In step 5-524 of Figure 5-5B, the user of client computer system 5-210 can utilize the received media content play list in conjunction with a media player application in order to cause client computer system 5-210 to transmit a request to web server 5-250 for delivery of desired media content, and wherein the user ID is automatically included therewith. The media content play list provided to client

computer system 5-210 by web server 5-250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to client computer system 5-250 and the order in which the content will be delivered. Additionally, the user of client computer system 5-250 can create one or more customized play lists and store those play lists in system 5-250 and/or within web server 5-250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions thereof. In another embodiment, the received media content play list can include a random media content delivery choice that the user of client computer system 5-210 can transmit to web server 5-250, with the user ID, to request delivery of the media content in a random manner.

In step 5-526, upon receiving the request for media content from client computer system 5-210, web server 5-250 determines whether the requesting media application operating on client computer system 5-210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 5-250 determines that the media application operating on system 5-210 is not a valid media application, the present embodiment proceeds to step 5-527 which in one embodiment, redirects client computer system 5-210 to a web site where the user of system 5-210 can download a valid media

player application or to a software application which can identify client computer system 5-210, log system 5-210 out of web server 5-250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 5-250 determines that the media application operating on system 5-210 is a valid media application, the present embodiment proceeds to step 5-528.

In step 5-528 of Figure 5-5B, the present embodiment causes web server 5-250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer system 5-210 is valid. If web server 5-250 determines that the user ID is invalid, the present embodiment proceeds to step 5-529 where client computer system 5-210 can be logged off web server 5-250 or client computer system 5-250 can be returned to step 5-506 (of Figure 5-5A) to re-register and to have another unique user ID generated by web server 5-250. It is noted that the order in which steps 5-526 and 5-528 are performed can be altered such that step 5-528 can be performed prior to step 5-526. If web server 5-250 determines that the user ID is valid, the present embodiment proceeds to step 5-530.

In step 5-530, prior to web server 5-250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 5-532, CCM 5-300 governs certain media player applications and/or functions thereof that are operable on client computer system 5-210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 5-300 can utilize system hooks 5-305 to accomplish the functionality of step 5-530.

In step 5-532 of Figure 5-5C, the present embodiment causes web server 5-250 to transmit to client computer system 5-210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 5-210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 5-251. The address is time sensitive because, in one embodiment, the content server 5-251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 5-251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 5-251 that are actively coupled to one or more client computer systems (e.g., 5-210 to 5-230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client computer systems are uncoupled from the media content source of content server 5-251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 5-210 to content server 5-251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 5-251), or by routing a requesting client computer system (e.g., 5-210, 5-220, or 5-230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 5-251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 5-210 to 5-230.

The functionality of step 5-532 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 5-210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

In step 5-534 of Figure 5-5C, upon receiving the redirection command, the present embodiment causes the media application operating on client computer system 5-210 to automatically transmit to content server 5-251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

In step 5-536 of Figure 5-5C, content server 5-251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 5-251 determines that the time sensitive access key is valid, the

present embodiment proceeds to step 5-538 of Figure 5-5C. However, if content server 5-251 determines that the time access key is not valid, the present embodiment proceeds to step 5-537, a client redirect.

In step 5-537, content server redirects client computer 5-210 to step 5-532 (not shown) where a new access key is generated. Alternatively, step 5-537 causes the present embodiment to return to step 5-504 of Figure 5-5A. In yet another embodiment, step 5-537 causes client computer system 5-210 to be disconnected from content server 5-251.

In step 5-538 of Figure 5-5C, content server 5-251 transmits the requested high fidelity media content to client computer system 5-210. It is noted that each media content file delivered to client computer system 5-210 can have a header attached thereto, prior to delivery, as described with reference to Figure 5-4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can be utilized.

Still referring to step 5-538, content server 5-251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 5-210 as fast as the network

transfer rate allows. Further, content server 5-251 can have its download rate adapted to be equal to the transfer rate of the network to which it is coupled. In another embodiment, the content server 5-251 download rate can be adapted to equal the network transfer rate of the client computer system 5-210 to which the media content is being delivered. For example, if client computer system 5-210 is coupled to Internet 5-201 via a T1 connection, then content server 5-251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 5-210, content server 5-251 is then able to transmit requested media content to another client computer system, e.g., 5-220 or 5-230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 5-250 to client computer system 5-210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 5-300 to client 5-210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 5-251 can transmit to client computer system 5-250 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 5-540 of Figure 5-5C, upon receiving the requested high fidelity media content from content server 5-251, the present embodiment causes client computer

system 5-210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content is stored in client computer system 5-210 in a manner that restricts unauthorized redistribution. For example, the present embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time.

Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory device, e.g., 5-103 or data storage device 5-108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is further noted that in one embodiment, when a user of client computer system 5-210 turns the computer off or causes client computer system 5-210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

Still referring to step 5-540, in another embodiment, the present embodiment can cause client computer system 5-210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media content can be stored in a manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is

noted that by storing media content with client computer system 5-210 (when allowed by applicable licensing agreements and copyright restrictions), content server 5-251 does not need to redeliver the same media content to client computer system 5-210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

In step 5-542 of Figure 5-5C, the received media content file is then fed into a media player application, which then runs it through a codec, e.g., coder/decoder 5-303 of CCM 5-300, in one embodiment. In response, coder/decoder 5-303 sends an authorization request to the server, e.g., 5-251, with attached authorization data, as described herein. In response to receiving codec's 5-303 authorization request, server 5-251 compares the received authorization data with that stored in server 5-251, and subsequently, the present embodiment proceeds to step 5-544.

In step 5-544, the server 5-251 responds with a pass or fail authorization. If server 5-251 responds with a fail, such that the received authorization data is invalid, the present method can proceed to step 5-545, where server 5-251 can, in one embodiment, notify the user of client system 5-210, e.g., by utilization of skin 5-306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented to the user of client computer system 5-210, thereby informing the user that the delivery failed. However, if the authorization data passes, the present method proceeds to step 5-546.

In step 5-546, server 5-251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file. In one embodiment, a decryption key can be included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 5-548.

In step 5-548 of Figure 5-5C, subsequent to media file decryption, the media file may be passed through CCM 5-300, e.g., a coder/decoder 5-303, to a media player application operating on client computer system 5-210 which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be required to experience the media content, e.g., skin 5-306 of Figure 5-3. Skin 5-306 may be necessary when CCM 5-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard media player can be utilized by client computer system 5-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player

application, e.g., in a frame by frame basis or in a buffer by buffer basis, coder/decoder 5-303 will repeatedly ensure that CCM 5-300 rules are being enforced at any particular moment during media playback, shown as step 5-550.

In step 5-550, as the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 5-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 5-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the present method proceeds to step 5-551. If the rules, in accordance with CCM 5-300, are enforced, the present method then proceeds to step 5-552.

In step 5-551, if the rules according to CCM 5-300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded.

In step 5-552, if the rules are enforced, in accordance with CCM 5-300, coder/decoder 5-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 5-210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the

present method as step 5-548. While the newly retrieved portion is presented, embodiments of the present method then again perform step 5-550, then step 5-552 or 5-551, then step 5-548, then 5-550, etc., in a continual loop until the media file contents are presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 5-300 can detect undesired activities and enforces those rules as defined by CCM 5-300.

Figure 5-6 is a diagram of an exemplary high-speed global media content delivery system 5-600, in accordance with one embodiment of the present invention. In one embodiment, system 5-600 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 5-210, 5-220, and/or 5-230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 5-600 includes a global delivery network 5-602 that can include multiple content servers, e.g., 5-604, 5-606, 5-608, 5-610, 5-612, 5-614, and 5-616, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 5-604 to 5-616 can store a portion, a substantial portion, or the entire contents of a media content library that can be delivered to client computer systems via a network, e.g., Internet 5-201, or a WAN (wide area network). Accordingly, each of content server 5-604 to 5-616 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

For example, a media delivery point (MDP) 5-616, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 5-451, to client computer systems within the Asiatic regions of the world while a media delivery point 5-612, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 5-604 to 5-616 represents the location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 5-604 to 5-616 can be located anywhere within the world, and as such are not limited to the cities shown in global network 5-602.

Still referring to Figure 5-6, it is further noted that global system 5-602 is described in conjunction with Figures 5-2, 5-3, 5-4, and 5-5, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 5-210 of Figure 5-2, interacting with a web server, e.g., web server 5-250 of Figure 5-2, as described herein, web server 5-250, in one embodiment, can redirect client computer system 5-210 to receive the desired media content from an MDP (e.g., 5-604 to 5-616) based on one or more differing criteria.

For example, computer system 5-210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 5-250 which can be located anywhere in the world. It is noted that steps 5-502 to 5-530 of Figure 5-5A and 5-5B

can then be performed as described herein such that the present embodiment proceeds to step 5-532 of Figure 5-5C. At step 5-532, the present embodiment can determine which media delivery points, e.g., 5-604, 5-606, 5-608, 5-610, 5-612, 5-614, or 5-616, can subsequently provide and deliver the desired media content to client computer system 5-210.

Still referring to Figure 5-6, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 5-210, e.g., media delivery point 5-616. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client computer system 5-210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 5-604 to 5-616) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 5-210 at step 5-532, web server 5-250 transmits to client computer system 5-210 a redirection command to media delivery point/content server 5-612 along with a time sensitive access key, also referred to as

a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 5-210 to eventually receive the requested media content. Within system 5-600, the redirection command can include a time sensitive address of the media content location within media delivery point 5-612. Accordingly, the New York City media delivery point 5-612 can subsequently provide and deliver the desired media content to client computer system 5-210. It is noted that steps 5-532 to 5-542 and step 5-537 of Figure 5-5C can be performed by media delivery point 5-512 in a manner similar to content server 5-251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 5-604 to 5-616, to provide high fidelity media content to client computer systems, e.g., 5-210 to 5-230, located throughout the world, communication network systems of the Internet 5-201 do not become overly congested. Additionally, global network 5-602 can deliver media content to a larger number of client computer systems (e.g., 5-210 to 5-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

Referring still to Figure 5-6, it is noted that media delivery points/content servers 5-604 to 5-616 of global network 5-602 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 5-604 to 5-616 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that media delivery points 5-604 to 5-616 can be functionally coupled such that if one of them fails, another media delivery point can

take over and fulfill its functionality. Additionally, one or more web servers similar to web server 5-250 can be coupled to global network 5-602 utilizing wired and/or wireless communication technologies.

Within system 5-600, content server/media delivery point 5-604 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 5-606 to 5-616 of global network 5-602 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 5-604.

Specifically, the web infrastructure of media delivery point 5-604 includes firewalls 5-618 and 5-620 which are each coupled to global network 5-602. Firewalls 5-618 and 5-620 can be coupled to global network 5-602 in diverse ways, e.g., utilizing wired and/or wireless communication technologies. Particularly, firewalls 5-618 and 5-620 can each be coupled to global network 5-602 via a 10/100 Ethernet handoff. However, system 5-600 is not limited in any fashion to this specific implementation. It is noted that firewalls 5-618 and 5-620 are implemented to prevent malicious users from accessing any part of the web infrastructure of media delivery point/content 5-604 in an unauthorized manner. Additionally, firewall 5-618 includes a device 5-636, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 5-640 coupled to device 5-636 while firewall 5-620 includes a device 5-638, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 5-642 coupled to device 5-638. Furthermore,

DB server 5-640 is coupled with device 5-638 and DB server 5-542 is coupled with device 5-536.

Still referring to Figure 5-6, and within media delivery point 5-604, firewall 5-618 is coupled to a director device 5-622 which is coupled to internal web application server 5-626 and 5-628, and a hub server 5-630. Firewall 5-620 is coupled to a director 5-624 which is coupled to internal web application servers 5-626 and 5-628, and hub server 5-630. Hub server 5-630 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 5-530 is coupled to a data storage device 5-632 capable of storing media content. Data storage device 5-632 can be implemented in a variety of ways, e.g., as a RAID (redundant array of independent disks) appliance.

It is noted that media delivery points 5-604 to 5-616 can be implemented in any manner similar to content server 5-250 described herein. Additionally, media delivery points 5-604 to 5-616 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 5-100 of Figure 5-1.

Advantageously, by providing a copyright compliance mechanism, e.g., 5-300, which can be easily and readily installed in a client computer system, e.g., 5-210, embodiments of the present invention can be implemented to control access to, control the delivery of, and control the user's experience with media content subject to copyright restrictions and licensing agreements, fore example, as defined by the

DMCA. Additionally, by closely associating a client computer system, e.g., 5-210, with the user thereof, and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

This application discloses a method of restricting client interaction of deliverable electronic media. In one embodiment, the method is comprised of detecting a media player application operable within a computer system. The media player application enables the computer system to present contents of a media file. The present method is further comprised of governing within said media player application a function that enables non-compliance with a usage restriction applicable to the media file. The present method is further comprised of controlling output of the media file. The controlling is performed by a compliance mechanism coupled to the computer system. The compliance mechanism is for enabling compliance with the usage restriction applicable to the media file.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 6: METHOD OF CONTROLLING RECORDING OF MEDIA

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals

capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop, handheld, and/or other portable electronic device. For purposes of the present application, the term “portable electronic device” is not intended to be limited solely to conventional handheld or portable computers. Instead, the term “portable electronic device” is also intended to include many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, and other personal digital devices.

Figure 6-1 is a block diagram illustrating an exemplary computer system 6-100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 6-100 can be nearly any type of computing system or electronic computing device including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 6-100 and which are executed by a processor(s) of computer system 6-100, in one embodiment. When executed, the instructions cause computer system 6-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 6-100 of Figure 6-1 comprises an address/data bus 6-110 for communicating information, one or more central processors 6-101 coupled to bus 6-110 for processing information and instructions. Central processor(s) 6-101 can

be a microprocessor or any alternative type of processor. Computer system 6-100 also includes a computer usable volatile memory 6-102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 6-110 for storing information and instructions for processor(s) 6-101. Computer system 6-100 further includes a computer usable non-volatile memory 6-103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc., coupled to bus 6-110 for storing static information and instructions for processor(s) 6-101. In one embodiment, non-volatile memory 6-103 can be removable.

System 6-100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 6-104 coupled to bus 6-110 for enabling computer 6-100 to interface with other electronic devices. Communication interface 6-104 can include wired and/or wireless communication functionality. For example, in one embodiment, communication interface 6-104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire (IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, a satellite link, an Internet feed, a cable modem, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device 6-104. In such an instance, communication interface 6-104 may include a DSL modem.

Computer 6-100 of Figure 6-1 can also include one or more computer usable data storage device(s) 6-108 coupled to bus 6-110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 6-108 can be a magnetic storage device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 6-108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks or random array of inexpensive discs) configuration. It is noted that data storage device 6-108 can be located internal and/or external of system 6-100 and communicatively coupled with system 6-100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 6-100. It is further noted that nearly any portable electronic device, e.g., device 6-100a, can also be communicatively coupled with system 6-100 via utilization of wired and/or wireless technology, thereby expanding the functionality of system 6-100.

System 6-100 can also include an optional display device 6-105 coupled to bus 6-110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 6-105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma display, a field emission display (FED) or any

other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 6-100 of Figure 6-1 further includes an optional alphanumeric input device 6-106 coupled to bus 6-110 for communicating information and command selections to processor(s) 6-101, in one embodiment. Alphanumeric input device 6-106 is coupled to bus 6-110 and includes alphanumeric and function keys. Also included in computer 6-100 is an optional cursor control device 6-107 coupled to bus 6-110 for communicating user input information and command selections to processor(s) 6-101. Cursor control device 6-107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, a optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 6-106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 6-100 is configured with such functionality.

Figure 6-2 is a block diagram of an exemplary network 6-200 in which embodiments of the present invention may be implemented. In one example, network 6-200 enables one or more authorized client computer systems (e.g., 6-210, 6-220, and 6-230), each of which are coupled to Internet 6-201, to receive media content from a media content server, e.g., 6-251, via the Internet 6-201 while

preventing unauthorized client computer systems from accessing media stored in a database of content server 6-251.

Network 6-200 includes a web server 6-250 and a content server 6-251 which are communicatively coupled to Internet 6-201. Further, web server 6-250 and content server 6-251 can be communicatively coupled without utilizing Internet 6-201, as shown. Web server 6-250, content server 6-251, and client computers 6-210, 6-220, and 6-230 can communicate with each other. It is noted that computers and servers of network 6-200 are well suited to be communicatively coupled in various implementations. For example, web server 6-250, content server 6-251, and client computer systems 6-210, 6-220, and 6-230 of network 6-200 can be communicatively coupled via wired communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 6-2, it is noted that web server 6-250, content server 6-251, and client computer systems 6-210, 6-220 and 6-230 of network 6-200 can, in one embodiment, be each implemented in a manner similar to computer system 6-100 of Figure 6-1. However, the server and computer systems in network 6-200 are not limited to such implementation. Additionally, web server 6-250 and content server 6-251 can perform various functionalities within network 6-200. It is also noted that, in one embodiment, web server 6-250 and content server 6-251 can both be disposed on a single or a plurality of physical computer systems, e.g., computer system 6-100 of Figure 6-1.

Further, it is noted that network 6-200 can operate with and deliver any type of media content, (e.g., audio, video, multimedia, graphics, information, data, software programs, etc.) in any format. In one example, content server 6-251 can provide audio and video files to client computers 6-210 to 6-230 via Internet 6-201.

Figure 6-3 is a block diagram of an exemplary copyright compliance mechanism (CCM) 6-300, for controlling distribution of, access to, and/or copyright compliance of media files, in accordance with an embodiment of the present invention. In one embodiment, CCM 6-300 contains one or more software components and instructions for enabling compliance with DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry association of America) licensing agreements regarding media files. In one embodiment, CCM 6-300 may be integrated into existing and/or newly developed media player and recorder applications. In another embodiment, CCM 6-300 may be implemented as stand alone but in conjunction with existing media player/recorder applications, such that CCM 6-300 is communicatively coupled to existing media player/recorder applications.

There are currently two types of copyright licenses recognized by the DMCA for the protection of broadcasted copyrighted material. One of the broadcast copyright licenses is a compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user

must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one example, the period of time can be three hours.

The other type of broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright holder grants permission for a server, e.g., web server 6-250 and/or content server 6-251 of Figure 6-2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with "no save" license, meaning that the end user is unable to save or store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user to save that particularly music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This is commonly referred to as an interactive with "save" license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

It is noted that the DMCA allows for the "perfect" reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording.

In one embodiment, CCM (copyright compliance mechanism) 6-300 can be stored in web server 6-250 and/or content server 6-251 of network 6-200 and is configured to be installed into each client computer system, e.g., 6-210, 6-220 and 6-230, enabled to access the media files stored within content server 6-251 and/or web server 6-250. Alternatively, copyright compliance mechanism 6-300 can be externally disposed and communicatively coupled with a client computer system 6-200 via, e.g., a portable media device 6-100a of Figure 6-1.

Copyright compliance mechanism 6-300 is configured to be operable while having portions of components, entire components, combinations of components, and/or comp, e.g., 6-210, 6-220, and/or 6-230.

Additionally, portions of components, entire components and/or combinations of components of CCM 6-300 can be readily updated, e.g., via Internet 6-201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or licensing agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications, or to counteract subversive and/or hacker-like attempts to unlawfully obtain one or more media files.

Referring to Figure 6-3, in one embodiment, CCM 6-300 is shown to include instructions 6-301 for enabling client computer system 6-210 to interact with web server 6-250 and content server 6-251 of network 6-200. Instructions 6-301 enable client computer system 6-210 to interact with servers, e.g., 6-250 and 6-251 in a network, e.g., 6-200.

The copyright compliance mechanism 6-300 also includes, in one embodiment, a user ID generator 6-302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 6-210. In one embodiment, the user ID and the cookie(s) are installed in computer system 6-210 prior to installation of the remaining components of the copyright compliance mechanism 6-300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 6-250 before the remaining components of a CCM 6-300 can be installed, within one embodiment of the present invention. Additionally, the user ID/user key can contain, but is not limited to, the user's name, the user's address, the user's credit card number, verified email address, and an identity (username) and password selected by the user. Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 6-210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 6-210, e.g., a MAC (machine address code) address and/or an IP address, and other information specific to the user and the computer system operated by the user. It is noted that the information regarding the client computer

system, e.g., 6-210, the user of system 6-210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 6-210, web server 6-250 can determine when a user of one computer system, e.g., 6-210, has given their username and password to another user using another computer system, e.g., 6-220. Because the username, password, and the user's computer system 6-210 are closely associated, web server 6-250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 6-250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 6-210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 6-250. Web server 6-250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 6-3, copyright compliance mechanism 6-300 further includes one or more coder/decoders (codec) 6-303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 6-300. In the present embodiment, coder/decoder 6-303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 6-251.

In the present embodiment, coder/decoder 6-303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 6-300 rules are being enforced at any particular moment during media playback. It is noted that differing coder/decoders 6-303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 6-3, copyright compliance mechanism 6-300 also includes one or more agent programs 6-304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 6-210, 6-220, or 6-230, a server, e.g., web server 6-250 and/or content server 6-251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 6-304 can also be configured to maintain system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 6-300 also includes one or more system hooks 6-305, in one embodiment of the present invention. A system hook 6-305 is, in one embodiment, a library that is installed in a computer system, e.g., 6-210, and intercepts system wide events. For example, a system hook 6-305, in conjunction with skins 6-306, can govern certain properties and/or functionalities of media player applications operating within the client computer system, e. g., 6-210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during playing of a media file, the progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that codec 6-303 compares the information for the media player application operating in client computer system, e.g., 6-210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a

media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 6-300 described herein.

The following C++ source code is exemplary implementation of the portion of a codec 6-303 for performing media player application detection, in accordance with an embodiment of the present invention.

```
int
IsRecorderPresent(TCHAR *    szAppClass,
                  TCHAR *    szProdName)
{
    TCHAR    szWndText[_MAX_PATH]; /* buffer to receive title string for
window */
    HWND     hWnd;                /* handle to target window for operation */
    int      nRetVal;             /* return value for operation */

    /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
        == 0)
    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
            != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
    else
    {
        /* attempt to locate window with specified class */
        if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
            != (HWND) 0)
        {
```

```

    {
        /* attempt to retrieve title string for window */
        if ( GetWindowText(hWnd,
                        szWndText,
                        _MAX_PATH)
            != 0)
        {
            /* attempt to locate product name within title string */
            if ( _tcsstr(szWndText, szProdName)
                != (TCHAR *) 0)
            {
                /* indicate application found */
                nRetVal = 1;
            }
        }
    }

    /* return to caller */
    return nRetVal;
}

```

It is further noted that codec 6-303 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 6-210. For example, codec 6-303, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 6-210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data pathways that are commonly used for recording, such that the utilized data pathway is governed by the copyright compliance mechanism 6-300. In one embodiment, this can be performed within a driver shim, e.g., wave driver shim 6-309 of Figures 6-5A and 6-5B.

A driver shim can be utilized for nearly any software output device, e.g., a standard Windows™ waveform output device, e.g., Windows™ Media Player, or

hardware output device, e.g., speakers or headphones. Client computer system 6-210 is configured such that the driver shim, (e.g., 6-309 of Figures 6-5A and 6-5B) will appear as the default waveform media device to client level application programs. Thus, requests for processing of waveform media input and/or output will pass through the driver shim prior to being forwarded to the actual waveform audio driver, media device driver 6-505 of Figures 6-5A and 6-5B. Such waveform input/output suppression can be triggered by other components of CCM 6-300, e.g., agent 6-304, to be active when a recording operation is initiated by a client computer system, e.g., 6-210, during the play back of media files which are subject to the DMCA.

It is noted that alternative driver shims can be implemented for nearly any waveform output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices, video, graphic and/or alphanumeric media files and video input and output devices.

The following C++ source code is an exemplary implementation of a portion of a codec 6-303 and/or a custom media device driver 6-307 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```
DWORD  
_stdcall  
widMessage(UINT      uDevId,
```

```

        UINT          uMsg,
        DWORD         dwUser,
        DWORD         dwParam1,
        DWORD         dwParam2)
{
    BOOL             bSkip;          /* flag indicating operation to be skipped */
    HWND             hWndMon;       /* handle to main window for monitor */
    DWORD            dwRetVal;      /* return value for operation */

    /* initialize variables */
    bSkip = FALSE;
    dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;

    if (uMsg == WIDM_START)
    {
        /* attempt to locate window for monitor application */
        if ( (hWndMon = FindMonitorWindow())
            != (HWND) 0)
        {
            /* obtain setting for driver */
            bDrvEnabled = ( SendMessages(hWndMon,
                                         uiRegMsg,
                                         0,
                                         0)
                          == 0)
                        ? FALSE : TRUE;
        }

        if (bDrvEnabled == TRUE)
        {
            /* indicate error in operation */
            dwRetVal = MMSYSERR_NOMEM;

            /* indicate operation to be skipped */
            bSkip = TRUE;
        }
    }

    if (bSkip == FALSE)
    {
        /* invoke entry point for original driver */
        dwRetVal = CallWidMessage(uDevId, uMsg, dwUser, dwParam1,
dwParam2);
    }

    /* return to caller */
    return dwRetVal;
}

```

It is noted that when properly configured, system hook 6-305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 6-210 to 6-230. In one embodiment, system hook 6-305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a Microsoft Windows™ operating environment, and accordingly can be implemented in a variety of ways. However, it is also noted that system hook 6-305 can be readily adapted for implementation in alternative operating systems, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 6-3, copyright compliance mechanism 6-300 also includes one or more skins 6-306, which can be designed to be installed in a client computer system, e.g., 6-210 to 6-230. In one embodiment, skins 6-306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 6-306 are customizable interfaces that, in one embodiment, are displayed on a display device (e.g., 6-105) of computer system 6-210 and provide functionalities for user interaction of delivered media content. Additionally, skins 6-306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

Furthermore, when system hook 6-305 is unable to govern a function of the media player application operable on a client computer system, e.g., 6-210, such that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 6-306 can be implemented to provide compliance.

Differing skins 6-306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 6-306a may be configured for utilization with a media content file protected under a non-interactive agreement (DMCA), such that skin 6-306a may not include a pause function, a stop function, a selector function, and/or a save function, etc. Another skin, e.g., skin 6-306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with "no save" agreement (DMCA), such that skin 6-306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with "save" agreement, a save or a burn to CD function.

Still referring to Figure 6-3, it is further noted that in the present embodiment, each skin 6-306 can have a unique name and signature. In one embodiment, skin 6-306 can be implemented, in part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 6-306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD5 hash, without knowledge of the MD5 hash table, changing the

name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance mechanism 6-300 verifies skin 6-306, MD5 hash tables advantageously provide a deterrent against modifications made to the skin.

In one embodiment, copyright compliance mechanism 6-300 also includes one or more custom media device driver(s) 6-307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 6-210, can be configured to utilize a custom media device application, e.g., custom media device 6-310 (shown in Figure 6-5B), to control unauthorized recording of media content files. A custom media device application can be, but is not limited to, a custom media audio device application for media files having sound content, a custom video device application for media files having graphical and/or alphanumeric content, etc. In one embodiment, custom media device 6-310 of Figure 6-5B is an emulation of the custom media device driver 6-307. With reference to audio media, the emulation is performed in a waveform audio driver associated with custom media device 6-310. Driver 6-307 is configured to receive a media file being outputted by system 6-210 prior to the media file being sent to a media output device, e.g., media output device 6-570, and/or a media output application, e.g., recording application 6-502. Examples of a media output device includes, but is not limited to, a video card for video files, a sound card for audio files, etc. Examples of a recording application can include, but is not limited to, CD burner applications for writing to another CDs, ripper applications which capture the media file and change the format of the media file, e.g., from a MP3 file to a .wav

file. In one embodiment, client computer system 6-210 is configured with a custom media device driver 6-307 emulating custom media device 6-310, and which is system 6-210's default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of skin 6-306 or a custom web based player application or as part of a CCM 6-300 installation bundle, for forcing or requiring system 6-210 to have driver 6-307 as the default driver.

Therefore, when a media content file is received by system 6-210 from server 6-251, the media content file is playable, provided the media content file passes through the custom media device application (e.g., 6-310 of Figure 6-5B), emulated by custom media device driver 6-307, prior to being outputted. However, if an alternative media player application is selected, delivered media files from server 6-251 will not play on system 6-210.

Thus, secured media player applications would issue a media request to the driver, e.g., 6-307, for the custom media device 6-310 which then performs necessary media input suppression, e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 6-307 to a Windows™ waveform audio driver operable on system 6-210, thus reducing instances of incompatibilities with existing

media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 6-300, e.g., agents 6-304, system hooks 6-305, and skins 6-306, or a combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of ways including, but limited to, those found at developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 6-210 being configured with a custom media device as the default device driver e.g., device 6-310 of Figures 6-5B and 6-5C, emulated by a custom media device driver 6-307, those media player applications that require their particular device driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured music. Further advantageous is that an emulated custom media device provides no native support for those media player applications used as a recording mechanism, e.g., DirectSound capture, (direct sound 6-504 of Figures 6-5A, 6-5B, and 6-5C) etc., that are able to bypass user-mode drivers for most media devices. Additionally, by virtue of the media content being sent through device driver 6-307, thus effectively disabling unauthorized saving/recording of media files, in one embodiment, media files that are delivered in a secured delivery system do not have to be encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-

encrypted media files utilizing less storage space and network resources than encrypted media files, networks having limited resources can utilize the functionalities of driver 6-307 of CCM 6-300 to provide compliance with copyright restrictions and/or licensing agreements applicable with a media content file without having the processing overhead of encrypted media files.

Figure 6-4 is an illustration of an exemplary system 6-400 for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention. Specifically, system 6-400 illustrates web server 6-250, content server 6-251, or a combination of web server 6-250 and content server 6-251 installing a copyright compliance mechanism (e.g., 6-300) in a client's computer system (e.g., 6-210) for controlling media file distribution and controlling user access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 6-210 can communicatively couple with a network (e.g., 6-200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 6-250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 6-250 can initiate a registration process with the requesting client 6-210. Client registration can be accomplished in a variety of ways. For example, web server 6-250 may deliver to a client 6-210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 6-250 including, but not

limited to, user's name, address, phone number, credit card number, verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 6-4, web server 6-250 can, in one embodiment, detect information related to the client's computer system, e.g., 6-210, and store that information in a user/media database 6-450. For example, web server 6-250 can detect a unique identifier of client computer system 6-210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network interface card) of client computer system 6-210 or the MAC address of the network interface adapter integrated on the motherboard of system 6-210. It is understood that a NIC enables a client computer system 6-210 to access web server 6-250 via Internet 6-201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 6-250 can, in one embodiment, detect and store (also in database 6-450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 6-210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 6-450) within web server 6-250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 6-210, all or part of this

information can be installed in client computer system 6-210. In one embodiment, client computer system 6-210 information can be in the form of a cookie. Web server 6-250 then verifies that the user and client computer system 6-210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 6-250 installs a copyright compliance mechanism (e.g., 6-300) into the client's computer system, e.g., 6-210, in one embodiment of the present invention. It is noted that web server 6-250 may not initiate installation of CCM 6-300 until the user ID, password, and client computer system 6-210 information is verified. A variety of common techniques can be employed to install an entire CCM 6-300, portions of components, entire components, and/or combinations or a function of components. For example, copyright compliance mechanism 6-300 can be installed in a hidden directory within client computer system 6-210, thereby preventing unauthorized access to it. In one embodiment of the present invention, it is noted that unless CCM 6-300 is installed in client computer system 6-210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 6-250 and/or content server 6-251,

Referring still to Figure 6-4, upon completion of client registration and installation of CCM 6-300, client computer system 6-210 can then request a media play list or a plurality of play lists, etc. In response, web server 6-250 determines whether the user of client computer system 6-210 is authorized to receive the media play list associated with the request. In one embodiment, web server 6-250 can request the username and password. Alternatively, web server 6-250 can utilize user database 6-450 to verify that computer 6-210 is authorized to receive a media

play list. If client computer 6-210 is not authorized, web server 6-250 can initiate client registration, as described herein. Additionally, web server 6-250 can disconnect computer 6-210 or redirect it to an alternative web site. Regardless, if the user and client computer system 6-210 are not authorized, web server 6-250 will not provide the requested play list to client computer system 6-210.

However, if client computer system 6-210 is authorized, web server 6-210 can check copyright compliance mechanism 6-300 within data base 6-450 to determine if it, or any of the components therein, have been updated since the last time client computer system 6-210 logged in to web server 6-250. If a component of CCM 6-300 has been updated, web server 6-250 can install the updated component and/or a more current version of CCM 6-300 into client computer system 6-210, e.g., via Internet 6-201. If CCM 6-300 has not been updated, web server 6-250 can then deliver the requested media play list to system 6-210 via Internet 6-201 along with an appended user key or user identification (ID). It is noted that user database 6-450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 6-210. Subsequently, the user of client computer system 6-210 can utilize the received media play list in combination with the media player application operating on system 6-210 to transmit a delivery request for one or more desired pieces of media content from web server 6-250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 6-4, upon receiving the media content delivery request, web server 6-250 can then check the validity of the requesting media application and

the attached user key. In one embodiment, web server 6-250 can utilize user database 6-450 to check their validity. If either or both are invalid, web server 6-250, in one embodiment, can redirect unauthorized client computer system 6-210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 6-300 verifies that skins 6-306 are installed in client computer system 6-210. Additionally, CCM 6-300 further verifies that system hook(s) 6-305 have been run or are running to govern certain functions of those media player applications operable within client computer system 6-210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 6-300 further diverts and/or redirects certain pathways that are commonly used for recording, e.g., driver 6-307 of Figure 6-5A, device 6-310 of Figure 6-5B, and device 6-570 of Figure 6-5C.. Once CCM 6-300 has performed the above described functions, web server 6-250 then, in one embodiment, issues to the client computer 6-210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 6-210 receiving the redirect command from web server 6-250, the media player application operating on client computer system 6-210 automatically transmits a new request and the time sensitive access key to content server 6-251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 6-251. If invalid, unauthorized client computer 6-210 is redirected by content server 6-250 to protect against abuse of the system and unauthorized access to

content server 6-251. If the time sensitive access key is valid, content server 6-251 retrieves the desired media content from content database 6-451 and delivers it to client computer system 6-210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 6-210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 6-210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 6-4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g., title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 6-300 that identifies the media file as originating from a content server 6-251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 6-300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or

the entire CCM 6-300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to modification of copyright compliance mechanism 6-300, modified portions of, or the entire updated CCM 6-300 can easily be installed in client computer system 6-210 in a variety of ways. For example, the updated CCM 6-300 can be installed during client interaction with web server 6-250, during user log-in, and/or while client computer system 6-210 is receiving the keyed play list.

Referring still to Figure 6-4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 6-251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be stored in a database 6-450, content database 6-451 or in some combination of databases 6-450 and 6-451. It is further noted that the messages being passed back and forth between client computer system 6-210 and web server 6-250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting with adds, public domain encryption programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be similarly encrypted. Furthermore, in another embodiment, the media

files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 6-307.

Subsequent to media file decryption, the media file may be passed through CCM 6-300, e.g., a coder/decoder 6-303, to a media player application operating on client computer system 6-210, e.g. playback application 6-501 of Figures 6-5A, 6-5B, 6-5C, and 6-6A, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 6-306 of Figure 6-3. A skin 6-306 may be necessary when CCM 6-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client computer system 6-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 6-303 will repeatedly ensure that CCM 6-300 rules are being enforced at any particular moment during media playback, shown as step 6-650 of Figure 6-6C.

As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 6-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 6-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These presentation stoppages are collectively shown as step 6-651 of Figure 6-6C.

If the rules, in accordance with CCM 6-300, are enforced, the codec/decoder 6-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 6-210. The newly retrieved portion of the media file is then presented by the client's media player application. While the newly retrieved portion is presented, CCM 6-300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 6-300 can detect undesired activities and enforces those rules as defined by CCM 6-300.

Figure 6-5A is an exemplary logic/bit path block diagram 6-500A showing utilization of a wave shim driver, e.g., wave shim driver 6-309 of Figure 6-3, in conjunction with copyright compliance mechanism 6-300, for selectively controlling recording of copyrighted media received by a client computer system, e.g., system 6-210, in one embodiment of the present invention. Copyright compliance mechanism 6-300 is, in one embodiment, installed and operational on client system 6-210 in the manner described herein.

In one embodiment, a copyright compliance mechanism 6-300 is shown as being communicatively coupled with a media playback application 6-501 via connection 6-520. Therefore, CCM 6-300 is enabled to communicate with playback application 6-501. In one embodiment, CCM 6-300 can be integrated into a media playback application. CCM 6-300 is also coupled to and controls a selectable switch 6-311 in wave shim driver 6-309 (as described in Figure 6-3) via connection 6-522. CCM 6-300 is further coupled to and controls a selectable switch 6-511 in direct sound 6-504 via connection 6-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., 6-499, CCM 6-300 controls whether switches 6-311 and 6-511 are open (shown), thus preventing incoming media 6-499 from reaching a media recording application, or closed (not shown) to allow recording of incoming media 6-499.

For example, incoming media 6-499 may originate from a content server, e.g., 6-251, coupled to system 6-210. In another example, incoming media 6-499 may

originate from a personal recording/electronic device, e.g., a MP3 player/recorder or similar device, coupled to system 6-210. Alternatively, incoming media 6-499 may originate from a magnetic, optical or alternative media storage device inserted into a media device player coupled to system 6-210, e.g., a CD or DVD inserted into a CD or DVD player, a hard disk in a hot swappable hard drive, an SD (secure digital card) inserted into a SD reader, and the like. In yet another example, incoming media 6-499 may originate from another media player application or media recording application. It is noted that incoming media 6-499 can originate from nearly any source that can be coupled to system 6-210. However, regardless of the source of incoming media 6-499, embodiments of the present invention, described herein, can prevent unauthorized recording of the media.

Figure 6-5A shows a media playback application 6-501, e.g., an audio, video, or other media player application, operable within system 6-210 and configured to receive incoming media 6-499. Playback application 6-501 can be a playback application provided by an operating system, e.g., Media Player for Windows™ by Microsoft, a freely distributed playback application downloadable from the Internet, e.g., RealPlayer or LiquidAudio, a playback application provided by a webcaster, e.g., PressPlay, or a playback application commercially available.

Figure 6-5A shows media device driver 6-505 which, in one implementation, may be a software driver for a sound card coupled to system 6-210 having a media output device 6-570, e.g., speakers or headphones, coupled therewith for media files having audio content. In another implementation, media device driver 6-505 may be

a software driver for a video card coupled with a display device, e.g., 6-105, for displaying media files having alphanumeric and/or graphical content, and so on. With reference to audio files, it is well known that a majority of recording applications assume a computer system, e.g., 6-210, has a sound card disposed therein, providing full-duplex sound functionality to system 6-210. This means media output driver 6-505 can simultaneously cause playback and recording of incoming media files 6-499. For example, media device driver 6-505 can playback media 6-499 along wave-out line 6-539 to media output device 6-570 (e.g., speakers for audible playback) via wave-out line 6-580 while outputting media 6-499 on wave-out line 6-540 to eventually reach recording application 6-502.

For purposes of Figures 6-5A, 6-5B, and 6-5C, the terms wave-in line and wave-out line are referenced from the perspective of media device driver 6-505. Additionally, for the most part, wave-in lines are downwardly depicted and wave-out lines are upwardly depicted in Figures 6-5A, 6-5B, and 6-5C.

Continuing with Figure 6-5A, playback application 6-501 is coupled with an operating system (O/S) multimedia subsystem 6-503 and direct sound 6-504 via wave-in lines 6-531 and 6-551 respectively. O/S multimedia subsystem 6-503 is coupled to a wave shim driver 6-309 via wave-in line 6-533 and wave-out line 6-546. O/S multimedia subsystem 6-503 is also coupled to a recording application 6-502 via wave-out line 6-548. Operating system (O/S) multimedia subsystem 6-503 can be any O/S multimedia subsystem, e.g., a Windows™ multimedia subsystem for system 6-210 operating under a Microsoft O/S, a QuickTime™ multimedia subsystem for

system 6-210 operating under an Apple O/S, and so on. Playback application 6-501 is also coupled with direct sound 6-504 via wave-in line 6-551.

Direct sound 6-504, in one instance, may represent access to a hardware acceleration feature in a standard audio device, enabling lower level access to components within media device driver 6-505. In another instance, direct sound 6-504 may represent a path that can be used by a recording application, e.g., Total Recorder, that can be further configured to bypass the default device driver, e.g., media device driver 6-505 to capture incoming media 6-499 for recording. For example, direct sound 6-504 can be enabled to capture incoming media 6-499 via wave-in line 6-551 and unlawfully output media 6-499 to a recording application 6-502 via wave-out line 6-568, as well as media 6-499 eventually going to media device driver 6-505, the standard default driver.

Still referring to Figure 6-5A, wave shim driver 6-309 is coupled with media device driver 6-505 via wave-in line 6-537 and wave-out line 6-542. Media device driver 6-505 is coupled with direct sound 6-504 via wave-in line 6-553 which is shown to converge with wave-in line 6-537 at media device driver 6-505. Media device driver 6-505 is also coupled with direct sound 6-504 via wave-out line 6-566.

Wave-out lines 6-542 and 6-566 are shown to diverge from wave-out line 6-540 at media device driver 6-505 into separate paths. Wave-out line 6-542 feeds into wave shim driver 6-309 and wave-out line 6-566 feeds into direct sound 6-504. When selectable switch 6-311 and 6-511 are open (shown), incoming media 6-499

cannot flow to recording application 6-502, thus preventing unauthorized recording of it.

For example, incoming media 6-499 is received at playback application 6-501. Playback application 6-501 activates and communicates to CCM 6-300 regarding copyright restrictions and/or licensing agreements applicable to incoming media 6-499. If recording restrictions apply to media 6-499, CCM 6-300 can, in one embodiment, open switches 6-311 and 6-511, thereby blocking access to recording application 6-502, effectively preventing unauthorized recording of media 6-499. In one embodiment, CCM 6-300 can detect if system 6-210 is configured with direct sound 6-504 selected as the default driver to capture incoming media 6-499, via wave-in line 6-551, or a recording application is detected and/or a hardware accelerator is active, such that wave driver shim 6-309 can be bypassed by direct sound 6-504. Upon detection, CCM 6-300 can control switch 6-511 such that the output path, wave-out line 6-568, to recording application 6-502 is blocked. It is further noted that CCM 6-300 can detect media recording applications and devices as described herein, with reference to Figure 6-3.

Alternatively, if media device driver 6-505 is selected as the default driver, incoming media 6-499 is output from playback application 6-501 to O/S multimedia subsystem 6-503 on wave-in line 6-531. From subsystem 6-503, media 6-499 is output to wave shim driver 6-309 via wave-in line 6-533. The wave shim driver 6-309 was described herein with reference to Figure 6-3. Media 6-499 is output from wave shim driver 6-309 to media device driver 6-505 via wave-in line 6-537. Once

received by media device driver 6-505, media 6-499 can be output via wave-out line 6-539 to a media output device 6-570 coupled therewith via wave-out line 6-580. Additionally, media device driver 6-505 can simultaneously output media 6-499 on wave-out line 6-540 back to wave shim driver 6-309. Dependent upon recording restrictions applicable to media 6-499, CCM 6-300 can, in one embodiment, close switch 6-311 (not shown as closed), thereby allowing media 6-499 to be output from wave shim driver 6-309 to subsystem 6-503 (via wave-out line 6-546) and then to recording application 6-502 via wave-out line 6-548. Alternatively, CCM 6-300 can also open switch 6-311, thereby preventing media 6-499 from reaching recording application 6-502.

It is particularly noted that by virtue of CCM 6-300 controlling both switches 6-311 and 6-511, and therefore controlling wave-out line 6-548 and wave-out line 6-568 leading into recording application 6-502, incoming media files, e.g., media 6-499, can be prevented from being recorded in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media. It is also noted that embodiments of the present invention in no way interfere with or inhibit the playback of incoming media 6-499.

Figure 6-5B is an exemplary logic/bit path block diagram 6-500B of a client computer system, e.g., 6-210, configured with a copyright compliance mechanism 6-300 for preventing unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 6-300 is, in

one embodiment, coupled with and operational on client system 6-210 in the manner with reference to Figures 6-4, 6-5A, 6-6, and 6-7.

Diagram 6-500B of Figure 6-5B is similar to diagram 6-500A of Figure 6-5A, with a few changes. Particularly, diagram 6-500B includes a custom media device 6-310 communicatively interposed between and coupled to O/S multimedia subsystem 6-503 and wave shim driver 6-309. Custom media device 6-310 is coupled to O/S multimedia subsystem via wave-in line 6-533 and wave-out line 6-546. Custom media device 6-310 is coupled with wave shim driver 6-309 via wave-in line 6-535 and wave-out line 6-544. Additionally, custom media device 6-310 is coupled with direct sound 6-504 via wave-in line 6-553 which converges with wave-in line 6-533 and wave-out line 6-566 which diverges from wave-out line 6-546, in one embodiment.

Also added to Figure 6-5B is a media hardware output device 6-570 that is coupled to media device hardware driver 6-505 via line 6-580. Media hardware output device 6-570 can be , but is not limited to, a sound card for audio playback, a video card for video, graphical, alphanumeric, etc, output, and the like.

In one embodiment, CCM 6-300 is communicatively coupled with playback application 6-501 via connection 6-520, waveform driver shim 6-309 via connection 6-522, and custom media device 6-310, via connection 6-521. CCM 6-300 is coupled to and controls a selectable switch 6-311 in waveform driver shim 6-309 via connection 6-522. CCM 6-300 is also coupled to and controls a selectable switch 6-

312 in custom audio device 6-310 via connection 6-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 6-499, CCM 6-300 controls whether switches 6-311 and 6-312 are open (shown), thus preventing the incoming media 6-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 6-499.

Continuing with Figure 6-5B, direct sound 6-504 is shown coupled with custom media device 6-310 via wave-in line 6-553, instead of being coupled with media device driver 6-505 (Figure 6-5A). In one embodiment, custom audio device 6-310 mandates explicit selection through system 6-210, meaning that custom audio device 6-310 needs to be selected as a default driver of system 6-210. By virtue of having the selection of custom media device 6-310 as the default driver of system 6-210, the data path necessary for direct sound 6-504 to capture the media content is selectively closed.

For example, incoming media 6-499 originating from nearly any source with reference to Figure 6-5A is received by media playback application 6-501 of system 6-210. Playback application 6-501 communicates to CCM 6-300, via connection 6-520, to determine whether incoming media 6-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 6-501 communicates with CCM 6-300 to control switch 6-311 and 6-312 accordingly. In the present example, recording of incoming media 6-499 would violate applicable restrictions and/or agreements and therefore switch 6-312 is in an open position, such that the

output path to recording application 6-502, e.g., wave-out line 6-548 and/or wave-out line 6-568, is effectively blocked, thereby preventing unauthorized recording of media 6-499.

Alternatively, if media device driver 6-505 is selected as the default driver, incoming media 6-499 continues from O/S multimedia subsystem 6-503, through custom audio device 6-310, wave driver shim 6-309, and into media device driver 6-505 where media 6-499 can be simultaneously output to media output device 6-570 via line 6-580, and output on wave-out line 6-540 to wave-and outputted by media device driver 6-505 to wave shim driver 6-309 on wave-out line 6-542. However, by virtue of CCM 6-300 controlling switch 6-311, wave-out line 6-544 which eventually leads to recording application 6-502 is blocked, thus effectively preventing unauthorized recording of media 6-499.

It is particularly noted that by virtue of CCM 6-300 controlling both switches 6-311 and 6-312 and therefore controlling wave-out line 6-548 and wave-out line 6-568, any incoming media files, e.g., incoming media 6-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 6-5B, it is further noted that custom media device 6-310 allows for unfettered playback of incoming media 6-499. Additionally, at any time during playback of media 6-499, custom media device 6-310 can be dynamically activated by CCM 6-300.

Figure 6-5C is an exemplary logic/bit path block diagram 6-500C of a client computer system, e.g., 6-210, configured with a copyright compliance mechanism 6-300 for preventing unauthorized output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 6-300 is, in one embodiment, coupled with and operational on client system 6-210 in the manner with reference to Figures 6-4, 6-5A, 6-5B, 6-6, and 6-7.

Diagram 6-500C of Figure 6-5C is similar to diagram 6-500B of Figure 6-5B, with a few changes. Particularly, diagram 6-500C includes a media hardware output device 6-570 that is coupled with a media device driver 6-505. In one embodiment, media hardware output device 6-570 can be a S/PDIF (Sony/Phillips Digital Interface) card for providing multiple outputs, e.g., an analog output 6-573 and a digital output 6-575. An alternative media hardware output device providing similar digital output can also be implemented as device 6-570 including, but not limited to, a USB (universal serial bus) output device and/or an externally accessible USB port located on system 6-210, a FireWire (IEEE1394) output device and/or an externally accessible FireWire port located on system 6-210, with wireline or wireless functionality. In the present embodiment, media hardware output device 6-570 is shown to include a switch 6-571 controlled by CCM 6-300 via communication line 6-523, similar to switches 6-311 and 6-312, for controlling output of incoming media 6-499.

In one embodiment, CCM 6-300 is communicatively coupled with playback application 6-501 via connection 6-520, waveform driver shim 6-309 via connection 6-522, custom media device 6-310, via connection 6-521, and media hardware output device 6-570 via connection 6-523. CCM 6-300 is coupled to and controls a selectable switch 6-311 in waveform driver shim 6-309 via connection 6-522. CCM 6-300 is also coupled to and controls a selectable switch 6-312 in custom audio device 6-310 via connection 6-521. CCM 6-300 is further coupled to and controls a selectable switch 6-571 in media hardware output device 6-570 via connection 6-523. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 6-499, CCM 6-300 controls whether switches 6-311 and 6-312 are open (shown), thus preventing the incoming media 6-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 6-499. Additionally, CCM 6-300 controls whether switch 6-571 is open (shown), thus preventing incoming media 6-499 from being output from digital output 6-575 of media hardware output device 6-570, or closed (not shown) to allow incoming media 6-499 to be output from media hardware output device 6-570.

By controlling media hardware output device 6-570, copyright compliance mechanism 6-300 can prevent unauthorized output of incoming media 6-499 to, e.g., a digital recording device that may be coupled with digital output 6-575 of media hardware output device 6-570. Accordingly, in one embodiment, CCM 6-300 is enabled to also detect digital recording devices that may be coupled to a digital output line, e.g., 6-571, of a media hardware output device, e.g., 6-570. Examples of a digital recording device that can be coupled to media hardware output device 6-

570 can include, but is not limited to, mini-disc recorders, MP3 recorders, personal digital recorders, digital recording devices coupled with multimedia systems, and/or nearly any digital device that can capture an incoming media 6-499 being output from a media hardware output device 6-570, e.g. a sound card.

Continuing with Figure 6-5C, direct sound 6-504 is shown coupled with custom media device 6-310 via wave-in line 6-553, instead of being coupled with media device driver 6-505 (Figure 6-5A). In one embodiment, custom audio device 6-310 mandates explicit selection through system 6-210, meaning that custom audio device 6-310 is needs to be selected as a default driver of system 6-210. By virtue of having the selection of custom media device 6-310 as the default driver of system 6-210, the data path necessary for direct sound 6-504 to capture the media content is selectively closed.

For example, incoming media 6-499 originating from nearly any source with reference to Figure 6-5A is received by media playback application 6-501 of system 6-210. Playback application 6-501 communicates to CCM 6-300, via connection 6-520, to determine whether incoming media 6-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 6-501 communicates with CCM 6-300 to control switch 6-311, 6-312, and 6-571 accordingly. In the present example, recording of incoming media 6-499 would violate applicable restrictions and/or agreements and therefore switch 6-312 is in an open position, such that the output path to recording application 6-502, e.g., wave-out line 6-548

and/or wave-out line 6-568, is effectively blocked, thereby preventing unauthorized recording of media 6-499.

Alternatively, if media device driver 6-505 is selected as the default driver, incoming media 6-499 continues from O/S multimedia subsystem 6-503, through custom audio device 6-310, wave driver shim 6-309, and into media device driver 6-505 where media 6-499 can be simultaneously output to media output device 6-570 via line 6-580, and output on wave-out line 6-540 to wave-and outputted by media device driver 6-505 to wave shim driver 6-309 on wave-out line 6-542. However, by virtue of CCM 6-300 controlling switch 6-311, wave-out line 6-544 which eventually leads to recording application 6-502 is blocked, thus effectively preventing unauthorized recording of media 6-499.

It is particularly noted that by virtue of CCM 6-300 controlling both switches 6-311 and 6-312 and therefore controlling wave-out line 6-548 and wave-out line 6-568, any incoming media files, e.g., incoming media 6-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 6-5C, it is particularly noted that although CCM 6-300 can prevent unauthorized recording of incoming media 6-499 by controlling switches 6-311 and 6-312, thus preventing incoming media 6-499 from reaching recording application 6-502, controlling switches 6-311 and 6-312 do nothing to prevent incoming media 6-499 from being captured by a peripheral digital device, e.g., a

mini-disc recorder, etc., coupled to a digital output 6-575 of device 6-570. Thus, by also controlling the output, via digital output 6-575 of media hardware output device 6-570, through control of switch 6-571, CCM 6-300 can prevent unauthorized capturing of incoming media 6-499 during output, e.g., on a sound card for audio files, a video card for video and/or graphical files, regardless of whether incoming media 6-499 is received in a secure and encrypted manner. However, when switch 6-571 is in a closed position, incoming media 6-499 may be played back in an unfettered manner. Additionally, at any time during playback of media 6-499, switch 6-312 of custom media device 6-310, switch 6-311 of media device driver 6-309, and/or switch 6-571 of media hardware output device 6-570 can be dynamically activated by CCM 6-300.

Figure 6-6A is an block diagram of a media file, e.g., incoming media 6-499, adapted to be received by a playback application, e.g., 6-501 of Figures 6-5A, 6-5B, and 6-5C, configured with an indicator 6-605 for enabling incoming media 6-499 to comply with rules according to the SCMS (serial copy management system). When applicable to a media file, e.g., 6-499, the SCMS allows for one copy of a copyrighted media file to be made, but not for copies of copies to be made. Thus, if incoming media 6-499 can be captured by a recording application, e.g., 6-502 of Figures 6-5A, 6-5B, and/or 6-5C, and/or a recording device, e.g. 6-529, and/or a peripheral recording device and/or a recording application coupled to a digital output of a media hardware output device, e.g., digital output 6-575 of media hardware output device 6-570 of Figures 6-5B and 6-5C, unauthorized copying and/or recording may be accomplished.

Playback application 6-501 is coupled with CCM 6-300 via communication line 6-520 in a manner analogous to Figures 6-5A, 6-5B and/or 6-5C. Although not shown in Figure 6-6, it is noted that CCM 6-300 is also coupled to switches 6-311 and 6-511 as shown in Figure 6-5A, switches 6-311 and 6-312 in Figure 6-5B, and switches 6-311, 6-312, and 6-571 in Figure 6-5C.

In one embodiment, an indicator 6-605 is attached to incoming media 6-499 for preventing unauthorized copying or recording in accordance with the SCMS. In one embodiment, indicator 6-605 can be a bit that may be transmitted prior to beginning the delivery of incoming media 6-499 to playback application 6-501. In another embodiment, indicator 6-605 may be placed at the beginning of the bit stream of incoming media 6-499. In another embodiment, indicator 6-605 may be placed within a frame period of incoming media 6-499, e.g., every fifth frame, or any other desired frame period. In another embodiment, indicator 6-605 may be transmitted at a particular time interval or intervals during delivery of the media file, e.g. incoming media 6-499. Thus, indicator 6-605 may be placed nearly anywhere within or attached to the bit stream related to incoming media 6-499.

Indicator 6-605 may be comprised of various indicators, e.g., a level 0 indicator, a level 1 indicator, and a level 2 indicator, in one embodiment of the present invention. In the present embodiment, a level 0 indicator may be for indicating to CCM 6-300 that copying is permitted without restriction, e.g., incoming media 6-499 is not copyrighted or that the copyright is not asserted. In the present

embodiment, a level 1 indicator may be for indicating to CCM 6-300 that one generation of copies of incoming media 6-499 may be made, such that incoming media 6-499 is an original copy and that one copy may be made. In the present embodiment, a level 2 indicator may be for indicating to CCM 6-300 that incoming media 6-499 is copyright protected and/or a copy thereof, and as such no digital copying is permitted.

For example, incoming media 6-499 is received by playback application 6-501. Application 6-501 detects an indicator 6-605 attached therewith, in this example, a level 2 bit is placed in the bit stream for indicating to CCM 6-300 that copying is not permitted.

For example, when CCM 6-300 is configured in system 6-210 such as that shown in Figure 6-5A, in response to a level 2 indicator bit, CCM 6-300, while controlling the audio path, then activates switches 6-311 and 6-511 to prevent any recording of incoming media 6-499.

When CCM 6-300 is configured in system 6-210 such as that shown in Figure 6-5B, in response to a level 2 indicator bit, CCM 6-300, while controlling the audio path, then activates switches 6-311 and 6-312 to prevent any recording of incoming media 6-499.

When CCM 6-300 is configured in system 6-210 such as that shown in Figure 6-5C, in response to a level 2 indicator bit, CCM 6-300, while controlling the audio

path, then activates switches 6-311, 6-312, and 6-571 to prevent any recording of incoming media 6-499.

It is noted that CCM 6-300 can activate or deactivate switches coupled therewith, as described herein with reference to Figures 6-5A, 6-5B, and 6-5c, thereby funneling incoming media 6-499 through the secure media path, in this instance the audio path, to prevent unauthorized copying of incoming media 6-499. It is further noted that CCM 6-300 can detect media recording applications and devices as described herein, with reference to Figure 6-3.

Figures 6-7A, 6-7B, and 6-7C, are a flowchart 6-700 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 6-700 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 6-104 and/or computer usable non-volatile memory 6-103 of Figure 6-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 6-700, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 6-7A, 6-7B, and 6-7C. Within the present embodiment, it should

be appreciated that the steps of flowchart 6-700 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for restricting recording of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 6-300, can be installed in the client computer system 6-210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 6-700 is described in conjunction with Figures 6-2, 6-3, 6-4, 6-5A, 6-5B, 6-5C, in order to more fully describe the operation of the present

embodiment. In step 6-702 of Figure 6-7A, a user of a computer system, e.g., 6-210, causes the computer to communicatively couple to a web server, e.g., 6-250, via one or more communication networks, e.g., Internet 6-201, and proceeds to attempt to log in. It is understood that the log in process of step 6-602 can be accomplished in a variety of ways in accordance with the present invention.

In step 6-704 of Figure 6-7A, web server 6-250 accesses a user database, e.g., 6-450, to determine whether the user and the computer system 6-210 logging in are registered with it. If the user and computer system 6-210 are registered with web server 6-250, the present embodiment proceeds to step 6-714. However, if the user and computer system 6-210 are logging in for the first time, web server 6-250 can initiate a user and computer system 6-210 registration process at step 6-706.

In step 6-706, registration of the user and computer system 6-210 is initiated. The user and computer system registration process can involve the user of computer system 6-210 providing personal information including, but not limited to, their name, address, phone number, credit card number, and the like. Web server 6-250 can verify the accuracy of the information provided. Web server 6-250 can also acquire information regarding the user's computer system 6-210 including, but not limited to, identification of media players disposed and operable on system 6-210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address.

Additionally, web server 6-250 can further request that the user of computer system 6-210 to select a username and password.

In step 6-708 of Figure 6-7A, subsequent to the completion of the registration process, web server 6-250 generates a unique user identification (ID) or user key associated with the user of client computer system 6-210. The unique user ID, or user key, is then stored by web server 6-250 in a manner that is associated with that registered user. Furthermore, one or more cookies containing that information specific to that user and the user's computer system 6-210, is installed in a non-volatile memory device, e.g., 6-103 and/or data storage device 6-108 of computer system 6-210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 6-210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user key, has been previously generated for the user and computer 6-210 that initially logged-in at step 6-702, the present embodiment proceeds to step 6-714

In step 6-710, web server 6-250 verifies that the user ID and the cookie(s) are properly installed in computer system 6-210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 6-250 can re-initiate the registration process at step 6-706. Alternatively, web server 6-250 can decouple computer system 6-210 from the network, thereby requiring a re-log in by the user of computer 6-210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 6-712.

In step 6-712 of Figure 6-7A, web server 6-250 can install a version of a copyright compliance mechanism, e.g., 6-300, onto one or more non-volatile memory devices of computer system 6-210. Installing CCM 6-300 into user's computer system 6-210 can facilitate client side compliance with licensing agreements and copyright restrictions applicable to specific delivered copyrighted media content. At step 6-712, the components of CCM 6-300, such as instructions 6-301, coder/decoder (codec) 6-303, agent programs 6-304, system hooks 6-305, skins 6-306, and custom media device drivers 6-307 (e.g., custom media device 6-310 of Figures 6-5B and 6-5C), are installed in computer system 6-210, such as that shown in Figures 6-5A, 6-5B, and 6-5C. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 6-300 into computer system 6-210. However, step 6-712 is well suited to install CCM 6-300 on computer system 6-210 in a wide variety of ways in accordance with the present embodiment. For example, CCM 6-300 can be installed as an integrated component within a media player application, media recorder application, and/or media player/recorder applications. Alternatively, CCM 6-300 can be installed as a stand alone mechanism within a client computer system 6-210. Additionally, CCM 6-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD, and/or as part of an installation package.

In step 6-714, web server 6-250 can request the previously established username and password of the user of client computer system 6-210. Accordingly, the user of client computer system 6-210 causes it to transmit to web server 6-250 the previously established username and password. Upon the receipt thereof, web

server 6-250 may access a user database, e.g., 6-450, to determine their validity. If the username and password are invalid, web server 6-250 refuses access wherein flowchart 6-500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 6-716.

In step 6-716 of Figure 6-7A, web server 6-250 can access media file database 6-450 to determine if copyright compliance mechanism 6-300 has been updated to reflect changes made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 6-300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 6-300 can provide compliance with current copyright restrictions.

Continuing with step 6-716, if web server 6-250 determines that CCM 6-300, or components thereof, of computer 6-210 has been updated, web server 6-250 initiates installation of the newer components and/or the most current version of CCM 6-300 into computer system 6-210, shown as step 6-718. If web server 6-250 determines that the current version of CCM 6-300 installed on system 6-210 does not have to be updated, the present embodiment proceeds to step 6-720 of Figure 6-7B.

In step 6-720 of Figure 6-7B, the user of client computer system 6-210 causes it to transmit to web server 6-250, e.g., via Internet 6-201, a request for a play list of available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g., 6-251.

In step 6-722, in response to web server 6-250 receiving the play list request, web server 6-250 transmits to client computer system 6-210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media content play list in a manner invisible to the user. It is noted that the media content in content server 6-251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, and the like. The media content play list of step 6-720 can be implemented in diverse ways. In one example, web server 6-250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media content titles, or different lists of titles, can be loaded from content server 6-251 and passed to a CGI (common gateway interface) program operating on web server 6-250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 6-210. It is understood that the CGI can be written in nearly any software computing language.

In step 6-724 of Figure 6-7B, the user of client computer system 6-210 can utilize the received media content play list in conjunction with a media player

application in order to cause client computer system 6-210 to transmit a request to web server 6-250 for delivery of desired media content, and wherein the user ID is automatically included therewith. The media content play list provided to client computer system 6-210 by web server 6-250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to client computer system 6-250 and the order in which the content will be delivered. Additionally, the user of client computer system 6-250 can create one or more customized play lists and store those play lists in system 6-250 and/or within web server 6-250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions thereof. In another embodiment, the received media content play list can include a random media content delivery choice, that the user of client computer system 6-210 can transmit to web server 6-250, with the user ID, to request delivery of the media content in a random manner.

In step 6-726, upon receiving the request for media content from client computer system 6-210, web server 6-250 determines whether the requesting media application operating on client computer system 6-210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 6-250 determines that the media application

operating on system 6-210 is not a valid media application, the present embodiment proceeds to step 6-727 which in one embodiment, redirects client computer system 6-210 to a web site where the user of system 6-210 can download a valid media player application or to a software application which can identify client computer system 6-210, log system 6-210 out of web server 6-250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 6-250 determines that the media application operating on system 6-210 is a valid media application, the present embodiment proceeds to step 6-728.

In step 6-728 of Figure 6-7B, the present embodiment causes web server 6-250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer system 6-210 is valid. If web server 6-250 determines that the user ID is invalid, the present embodiment proceeds to step 6-729 where client computer system 6-210 can be logged off web server 6-250 or client computer system 6-250 can be returned to step 6-706 (of Figure 6-7A) to re-register and to have another unique user ID generated by web server 6-250. It is noted that the order in which steps 6-726 and 6-728 are performed can be altered such that step 6-728 can be performed prior to step 6-726. If web server 6-250 determines that the user ID is valid, the present embodiment proceeds to step 6-730.

In step 6-730, prior to web server 6-250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 6-732, CCM 6-300 governs certain media player applications and/or functions thereof that are

operable on client computer system 6-210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 6-300 can utilize system hooks 6-305 to accomplish the functionality of step 6-730.

In step 6-732 of Figure 6-7C, the present embodiment causes web server 6-250 to transmit to client computer system 6-210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 6-210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 6-251. The address is time sensitive because, in one embodiment, the content server 6-251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 6-251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 6-251 that are actively coupled to one or more client computer systems (e.g., 6-210 to 6-230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client

computer systems are uncoupled from the media content source of content server 6-251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 6-210 to content server 6-251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 6-251), or by routing a requesting client computer system (e.g., 6-210, 6-220, or 6-230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 6-251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 6-210 to 6-230.

The functionality of step 6-732 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 6-210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

In step 6-734 of Figure 6-7C, upon receiving the redirection command, the present embodiment causes the media playback application 6-501 (Figures 6-5A, 6-5B, and 6-5C) operating on client computer system 6-210 to automatically transmit to content server 6-251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

In step 6-736 of Figure 6-7C, content server 6-251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 6-251 determines that the time sensitive access key is valid, the present embodiment proceeds to step 6-738 of Figure 6-7C. However, if content server 6-251 determines that the time access key is not valid, the present embodiment proceeds to step 6-737, a client redirect.

In step 6-737, content server redirects client computer 6-210 to step 6-732 (not shown) where a new access key is generated. Alternatively, step 6-737 causes the present embodiment to return to step 6-704 of Figure 6-7A. In yet another embodiment, step 6-737 causes client computer system 6-210 to be disconnected from content server 6-251.

In step 6-738 of Figure 6-7C, content server 6-251 transmits the requested high fidelity media content to client computer system 6-210. It is noted that each media content file delivered to client computer system 6-210 can have a header attached thereto, prior to delivery, as described with reference to Figure 6-4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can be utilized.

Still referring to step 6-738, content server 6-251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 6-210 as fast as the network transfer rate allows. Further, content server 6-251 can have its download rate adapted to be equal to the transfer rate of the network to which it is coupled. In another embodiment, the content server 6-251 download rate can be adapted to equal the network transfer rate of the client computer system 6-210 to which the media content is being delivered. For example, if client computer system 6-210 is coupled to Internet 6-201 via a T1 connection, then content server 6-251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 6-210, content server 6-251 is then able to transmit requested media content to another client computer system, e.g., 6-220 or 6-230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 6-250 to client computer system 6-210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 6-300 to client 6-210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 6-251 can transmit to client computer system 6-250 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 6-740 of Figure 6-7C, upon receiving the requested high fidelity media content from content server 6-251, the present embodiment causes client computer system 6-210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content is stored in client computer system 6-210 in a manner that restricts unauthorized redistribution. For example, the present embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time. Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory device, e.g., 6-103 or data storage device 6-108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is further noted that in one embodiment, when a user of client computer system 6-210 turns the computer off or causes client computer system 6-210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

Still referring to step 6-740, in another embodiment, the present embodiment can cause client computer system 6-210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media content can be stored in a

manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is noted that by storing media content with client computer system 6-210 (when allowed by applicable licensing agreements and copyright restrictions), content server 6-251 does not need to redeliver the same media content to client computer system 6-210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

In step 6-742 of Figure 6-7C, the received media content file is then fed into a media player application (e.g., playback application 6-501 of Figures 6-5A, 6-5B, and 6-5C), which then runs it through a codec, e.g., coder/decoder 6-303 of CCM 6-300, in one embodiment. In response, coder/decoder 6-303 sends an authorization request to the server, e.g., 6-251, with attached authorization data, as described herein. In response to receiving codec's 6-303 authorization request, server 6-251 compares the received authorization data with that stored in server 6-251, and subsequently, the present embodiment proceeds to step 6-744.

In step 6-744, the server 6-251 responds with a pass or fail authorization. If server 6-251 responds with a fail, such that the received authorization data is invalid, the present method can proceed to step 6-745, where server 6-251 can, in one embodiment, notify the user of client system 6-210, e.g., by utilization of skin 6-306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented

to the user of client computer system 6-210, thereby informing the user that the delivery failed. However, if the authorization data passes, the present method proceeds to step 6-746.

In step 6-746, server 6-251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file via media playback application 6-501 of Figures 6-5A, 6-5B, and 6-5C. In one embodiment, a decryption key can be included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 6-748.

In step 6-748 of Figure 6-6C, subsequent to media file decryption, the media file may be passed through CCM 6-300, e.g., a coder/decoder 6-303, to a media player application operating on client computer system 6-210, e.g., playback application 6-501 of Figures 6-5A, 6-5B, and 6-5C, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be involved in order to experience the media content, e.g., skin 6-306 of Figure 6-3. Skin 6-306 may be implemented when CCM 6-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard

media player can be utilized by client computer system 6-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, coder/decoder 6-303 will repeatedly ensure that CCM 6-300 rules are being enforced at any particular moment during media playback, shown as step 6-750.

In step 6-750, as the media file content is delivered to the media player application, e.g., media player application 6-501 of Figures 6-5A, 6-5B, and 6-5C, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 6-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 6-300. If the rules are not enforced, e.g., change due to a user opening up a recording application (e.g., Total Recorder or alternative application) the present method proceeds to step 6-751. If the rules, in accordance with CCM 6-300, are enforced, the present method then proceeds to step 6-752.

In step 6-751 of Figure 6-7C, if the rules according to CCM 6-300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In one embodiment, CCM 6-300 can selectively control switches 6-311 and 6-511 (Figure 6-5A) to prevent output of incoming media 6-499 (Figures 6-5A, 6-

5B, and 6-5C) to a recording application 6-502 (Figures 6-5A, 6-5B, and 6-5C, via wave shim driver 6-309 and direct sound 6-504 respectively, thus preventing unauthorized recording of incoming media 6-499. In another embodiment, CCM 6-300 can selectively control switches 6-311 and 6-312 (Figure 6-5B) to prevent output of incoming media 6-499 to recording application 6-502 via wave shim driver 6-309 and custom media device 6-310, thus preventing unauthorized recording of incoming media 6-499. In yet another embodiment, CCM 6-300 can selectively control switches 6-311, 6-312, to not only prevent incoming media 6-499 from being recorded in an unauthorized manner but can also selectively control switch 6-571 (Figure 6-5C) to prevent unauthorized output of incoming media 6-499 via digital output 6-575 of media hardware output device 6-570. In one embodiment, incoming media 6-499 may not be output from digital output 6-575. In another embodiment, incoming media 6-499 may be output via digital output 6-575 but in an inaudible manner, e.g., silence. In yet another embodiment, incoming media 6-499 be audible but recording functionality can be disabled, such that the media content cannot be recorded.

In step 6-752, if the rules are enforced in accordance with CCM 6-300, coder/decoder 6-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 6-210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the present method as step 6-748. While the newly retrieved portion is presented, embodiments of the present method then again perform step 6-750, then step 6-752 or 6-751, then step 6-748, then 6-750, etc., in a continual loop until the media file

contents are presented in their entirety. Advantageously, by constantly monitoring playing media files, CCM 6-300 can detect undesired activities and enforce those rules defined by CCM 6-300.

Figure 6-8 is a diagram of an exemplary high-speed global media content delivery system 6-800, in accordance with one embodiment of the present invention. In one embodiment, system 6-800 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 6-210, 6-220, and/or 6-230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 6-800 includes a global delivery network 6-802 that can include multiple content servers, e.g., 6-804, 6-806, 6-808, 6-810, 6-812, 6-814, and 6-816, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 6-804 to 6-816 can store a portion, a substantial portion, or the entire contents of a media content library that can be delivered to client computer systems via a network, e.g., Internet 6-201, or a WAN (wide area network). Accordingly, each of content server 6-804 to 6-816 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

For example, a media delivery point (MDP) 6-816, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 6-451, to client computer systems within the Asiatic regions

of the world while a media delivery point 6-812, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 6-804 to 6-816 represents the location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 6-804 to 6-816 can be located anywhere within the world, and as such are not limited to the cities shown in global network 6-802.

Still referring to Figure 6-8, it is further noted that global system 6-802 is described in conjunction with Figures 6-2, 6-3, 6-4, 6-5, and 6-6, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 6-210 of Figure 6-2, interacting with a web server, e.g., web server 6-250 of Figure 6-2, as described herein, web server 6-250, in one embodiment, can redirect client computer system 6-210 to receive the desired media content from an MDP (e.g., 6-804 to 6-816) based on one or more differing criteria.

For example, computer system 6-210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 6-250 which can be located anywhere in the world. It is noted that steps 6-702 to 6-730 of Figures 6-7A and 6-7B can then be performed as described herein such that the present embodiment proceeds to step 6-732 of Figure 6-7C. At step 6-732, the present embodiment can

determine which media delivery points, e.g., 6-804, 6-806, 6-808, 6-810, 6-812, 6-814, or 6-816, can subsequently provide and deliver the desired media content to client computer system 6-210.

Still referring to Figure 6-8, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 6-210, e.g., media delivery point 6-816. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client computer system 6-210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 6-804 to 6-816) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 6-210 at step 6-732, web server 6-250 transmits to client computer system 6-210 a redirection command to media delivery point/content server 6-812 along with a time sensitive access key, also referred to as a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 6-210 to eventually receive the requested media content.

Within system 6-800, the redirection command can include a time sensitive address of the media content location within media delivery point 6-812. Accordingly, the New York City media delivery point 6-812 can subsequently provide and deliver the desired media content to client computer system 6-210. It is noted that steps 6-732 to 6-742 and step 6-737 of Figure 6-7C can be performed by media delivery point 6-812 in a manner similar to content server 6-251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 6-804 to 6-816, to provide high fidelity media content to client computer systems, e.g., 6-210 to 6-230, located throughout the world, communication network systems of the Internet 6-201 do not become overly congested. Additionally, global network 6-802 can deliver media content to a larger number of client computer systems (e.g., 6-210 to 6-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 6-320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

Referring still to Figure 6-8, it is noted that media delivery points/content servers 6-804 to 6-816 of global network 6-802 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 6-804 to 6-816 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that media delivery points 6-804 to 6-816 can be functionally coupled such that if one of them fails, another media delivery point can take over and fulfill its functionality. Additionally, one or more web servers similar to

web server 6-250 can be coupled to global network 6-802 utilizing wired and/or wireless communication technologies.

Within system 6-800, content server/media delivery point 6-804 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 6-806 to 6-816 of global network 6-802 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 6-804.

Specifically, the web infrastructure of media delivery point 6-804 includes firewalls 6-818 and 6-820 which are each coupled to global network 6-802. Firewalls 6-818 and 6-820 can be coupled to global network 6-802 in diverse ways, e.g., utilizing wired and/or wireless communication technologies. Particularly, firewalls 6-818 and 6-820 can each be coupled to global network 6-702 via a 10/100 Ethernet handoff. However, system 6-800 is not limited in any fashion to this specific implementation. It is noted that firewalls 6-818 and 6-820 are implemented to prevent malicious users from accessing any part of the web infrastructure of media del836, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 6-840 coupled to device 6-836 while firewall 6-820 includes a device 6-838, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 6-842 coupled to device 6-838. Furthermore, DB server 6-840 is coupled with device 6-838 and DB server 6-842 is coupled with device 6-836.

Still referring to Figure 6-8, and within media delivery point 6-804, firewall 6-818 is coupled to a director device 6-822 which is coupled to internal web application server 6-826 and 6-828, and a hub server 6-830. Firewall 6-820 is coupled to a director 6-824 which is coupled to internal web application servers 6-826 and 6-828, and hub server 6-830. Hub server 6-830 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 6-780 is coupled to a data storage device 6-832 capable of storing media content. Data storage device 6-832 can be implemented in a variety of ways, e.g., as a RAID (redundant array of inexpensive/independent disks) appliance.

It is noted that media delivery points 6-804 to 6-816 can be implemented in any manner similar to content server 6-250 described herein. Additionally, media delivery points 6-804 to 6-816 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 6-100 of Figure 6-1.

Advantageously, by providing a copyright compliance mechanism, e.g., 6-300, which can be easily and readily installed in a client computer system, e.g., 6-210, embodiments of the present invention can be implemented to control access to, control the delivery of, and control the user's experience with media content subject to copyright restrictions and licensing agreements, for example, as defined by the DMCA. Additionally, by closely associating a client computer system, e.g., 6-210, with the user thereof, and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

A method of preventing unauthorized recording of electronic media is described. The method is comprised of activating a compliance mechanism in response to receiving media content by a client system. The compliance mechanism is coupled to the client system. The media content presentation application is operable and coupled to the compliance mechanism. The method is further comprised of controlling a data output path of the client computer with the compliance mechanism. The method is further comprised of directing the media content via the data output path to a custom media device for selectively restricting output of the media content. The custom media device is coupled to the compliance mechanism and to the media content presentation application. The method is further comprised of preventing a recording application coupled to the client computer system from recording the media content file when recording violates usage restriction applicable to the media content.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 7: METHOD FOR REDIRECTING OF KERNAL DATA PATH
FOR CONTROLLING RECORDING OF MEDIA

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For

reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop, handheld, and/or other portable electronic device. For purposes of the present application, the term "portable electronic device" is not intended to be limited solely to conventional handheld or portable computers. Instead, the term "portable electronic device" is also intended to include

many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, satellite radios, and other personal digital devices.

Figure 7-1 is a block diagram illustrating an exemplary computer system 7-100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 7-100 can be nearly any type of computing system or electronic computing device including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 7-100 and which are executed by a processor(s) of computer system 7-100, in one embodiment. When executed, the instructions cause computer system 7-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 7-100 of Figure 7-1 comprises an address/data bus 7-110 for communicating information, one or more central processors 7-101 coupled to bus 7-110 for processing information and instructions. Central processor(s) 7-101 can be a microprocessor or any alternative type of processor. Computer system 7-100 also includes a computer usable volatile memory 7-102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 7-110 for storing information and instructions for processor(s) 7-101. Computer system 7-100 further includes a computer usable non-volatile memory 7-103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc.,

coupled to bus 7-110 for storing static information and instructions for processor(s) 7-101. In one embodiment, non-volatile memory 7-103 can be removable.

System 7-100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 7-104 coupled to bus 7-110 for enabling computer 7-100 to interface with other electronic devices. Communication interface 7-104 can include wired and/or wireless communication functionality. For example, in one embodiment, communication interface 7-104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire (IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, a satellite link, an Internet feed, a cable modem, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device 7-104. In such an instance, communication interface 7-104 may include a DSL modem.

Computer 7-100 of Figure 7-1 can also include one or more computer usable data storage device(s) 7-108 coupled to bus 7-110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 7-108 can be a magnetic storage device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 7-108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks or random array of inexpensive discs) configuration. It is noted that data storage device 7-108

can be located internal and/or external of system 7-100 and communicatively coupled with system 7-100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 7-100. It is further noted that nearly any portable electronic device, e.g., device 7-100a, can also be communicatively coupled with system 7-100 via utilization of wired and/or wireless technology, thereby expanding the functionality of system 7-100.

System 7-100 can also include an optional display device 7-105 coupled to bus 7-110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 7-105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma display, a field emission display (FED) or any other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 7-100 of Figure 7-1 further includes an optional alphanumeric input device 7-106 coupled to bus 7-110 for communicating information and command selections to processor(s) 7-101, in one embodiment. Alphanumeric input device 7-106 is coupled to bus 7-110 and includes alphanumeric and function keys. Also included in computer 7-100 is an optional cursor control device 7-107 coupled to bus 7-110 for communicating user input information and command selections to processor(s) 7-101. Cursor control device 7-107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, an optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 7-106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 7-100 is configured with such functionality.

Figure 7-2 is a block diagram of an exemplary network 7-200 in which embodiments of the present invention may be implemented. In one example, network 7-200 enables one or more authorized client computer systems (e.g., 7-210, 7-220, and 7-230), each of which are coupled to Internet 7-201, to receive media content from a media content server, e.g., 7-251, via the Internet 7-201 while preventing unauthorized client computer systems from accessing media stored in a database of content server 7-251.

Network 7-200 includes a web server 7-250 and a content server 7-251 which are communicatively coupled to Internet 7-201. Further, web server 7-250 and content server 7-251 can be communicatively coupled without utilizing Internet 7-201, as shown. Web server 7-250, content server 7-251, and client computers 7-210, 7-220, and 7-230 can communicate with each other. It is noted that computers and servers of network 7-200 are well suited to be communicatively coupled in various implementations. For example, web server 7-250, content server 7-251, and client computer systems 7-210, 7-220, and 7-230 of network 7-200 can be communicatively coupled via wired communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 7-2, it is noted that web server 7-250, content server 7-251, and client computer systems 7-210, 7-220 and 7-230 of network 7-200 can, in one embodiment, be each implemented in a manner similar to computer system 7-100 of Figure 7-1. However, the server and computer systems in network 7-200 are not limited to such implementation. Additionally, web server 7-250 and content server 7-251 can perform various functionalities within network 7-200. It is also noted that, in one embodiment, web

server 7-250 and content server 7-251 can both be disposed on a single or a plurality of physical computer systems, e.g., computer system 7-100 of Figure 7-1.

Further, it is noted that network 7-200 can operate with and deliver any type of media content, (e.g., audio, video, multimedia, graphics, information, data, software programs, etc.) in any format. In one example, content server 7-251 can provide audio and video files to client computers 7-210 to 7-230 via Internet 7-201.

Figure 7-3 is a block diagram of an exemplary copyright compliance mechanism (CCM) 7-300, for controlling distribution of, access to, and/or copyright compliance of media files, in accordance with an embodiment of the present invention. In one embodiment, CCM 7-300 contains one or more software components and instructions for enabling compliance with DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry association of America) licensing agreements regarding media files. In one embodiment, CCM 7-300 may be integrated into existing and/or newly developed media player and recorder applications. In another embodiment, CCM 7-300 may be implemented as stand alone but in conjunction with existing media player/recorder applications, such that CCM 7-300 is communicatively coupled to existing media player/recorder applications. Alternatively, CCM 7-300 can be installed as a stand alone mechanism within a client computer system 7-210. Additionally, CCM 7-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD (secure digital card), and/or as part of an installation package. In another example, CCM 7-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD, reading a document, viewing a video,

etc. It is noted that, in one embodiment, CCM 7-300 may be installed on client system 7-210 in a clandestine manner, relative to a user.

There are currently two types of copyright licenses recognized by the DMCA for the protection of broadcasted copyrighted material. One of the broadcast copyright licenses is a compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one example, the period of time can be three hours.

The other type of broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright holder grants permission for a server, e.g., web server 7-250 and/or content server 7-251 of Figure 7-2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with "no save" license, meaning that the end user is unable to save or store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user to save that particularly music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This

is commonly referred to as an interactive with “save” license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

It is noted that the DMCA allows for the “perfect” reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording.

In one embodiment, CCM (copyright compliance mechanism) 7-300 can be stored in web server 7-250 and/or content server 7-251 of network 7-200 and is configured to be installed into each client computer system, e.g., 7-210, 7-220 and 7-230, enabled to access the media files stored within content server 7-251 and/or web server 7-250. Alternatively, copyright compliance mechanism 7-300 can be externally disposed and communicatively coupled with a client computer system 7-200 via, e.g., a portable media device 7-100a of Figure 7-1. In yet another embodiment, CCM 7-300 can be configured to be operable from a media storage device upon which media files may be disposed.

Copyright compliance mechanism 7-300 is configured to be operable while having portions of components, entire components, combinations of components, disposed within one or more memory units and/or data storage devices of a computer system, e.g., 7-210, 7-220, and/or 7-230.

Additionally, portions of components, entire components and/or combinations of components of CCM 7-300 can be readily updated, e.g., via Internet 7-201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or

licensing agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications, or to counteract subversive and/or hacker-like attempts to unlawfully obtain one or more media files.

Referring to Figure 7-3, in one embodiment, CCM 7-300 is shown to include instructions 7-301 for enabling client computer system 7-210 to interact with web server 7-250 and content server 7-251 of network 7-200. Instructions 7-301 enable client computer system 7-210 to interact with servers, e.g., 7-250 and 7-251 in a network, e.g., 7-200.

The copyright compliance mechanism 7-300 also includes, in one embodiment, a user ID generator 7-302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 7-210. In one embodiment, the user ID and the cookie(s) are installed in computer system 7-210 prior to installation of the remaining components of the copyright compliance mechanism 7-300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 7-250 before the remaining components of a CCM 7-300 can be installed, within one embodiment of the present invention. Additionally, the user ID/user key can contain, but is not limited to, the user's name, the user's address, the user's credit card number, verified email address, and an identity (username) and password selected by the user. Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 7-210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 7-210, e.g., a MAC (machine address code) address and/or an IP address, and other information specific to the user and the computer system operated by the user. It is noted that the information regarding

the client computer system, e.g., 7-210, the user of system 7-210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 7-210, web server 7-250 can determine when a user of one computer system, e.g., 7-210, has given their username and password to another user using another computer system, e.g., 7-220. Because the username, password, and the user's computer system 7-210 are closely associated, web server 7-250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 7-250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 7-210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 7-250. Web server 7-250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 7-3, copyright compliance mechanism 7-300 further includes one or more coder/decoders (codec) 7-303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 7-300. In the present embodiment, coder/decoder 7-303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 7-251.

In the present embodiment, coder/decoder 7-303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player

application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 7-300 rules are being enforced at any particular moment during media playback. It is noted that differing coder/decoders 7-303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 7-3, copyright compliance mechanism 7-300 also includes one or more agent programs 7-304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 7-210, 7-220, or 7-230, a server, e.g., web server 7-250 and/or content server 7-251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 7-304 can also be configured to maintain system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 7-300 also includes one or more system hooks 7-305, in one embodiment of the present invention. A system hook 7-305 is, in one embodiment, a library that is installed in a computer system, e.g., 7-210, and intercepts system wide events. For example, a system hook 7-305, in conjunction with skins 7-306, can

govern certain properties and/or functionalities of media player applications operating within the client computer system, e. g., 7-210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during playing of a media file, the progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that codec 7-303 compares the information for the media player application operating in client computer system, e.g., 7-210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 7-300 described herein.

The following C++ source code is exemplary implementation of the portion of a codec 7-303 for performing media player application detection, in accordance with an

embodiment of the present invention. In another embodiment, the following source code can be modified to detect kernel streaming mechanisms operable within client system 7-210.

```

int
IsRecorderPresent(TCHAR * szAppClass,
                  TCHAR *  szProdName)
{
    TCHAR          szWndText[_MAX_PATH]; /* buffer to receive title string for
window */
    HWND           hWnd;                 /* handle to target window for operation */
    int            nRetVal;              /* return value for operation */

    /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
        == 0)
    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
            != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
    else
    {
        /* attempt to locate window with specified class */
        if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
            != (HWND) 0)
        {
            /* attempt to retrieve title string for window */
            if ( GetWindowText(hWnd,
                              szWndText,
                              _MAX_PATH)
                != 0)
            {
                /* attempt to locate product name within title string */
                if ( _tcsstr(szWndText, szProdName)
                    != (TCHAR *) 0)
                {
                    /* indicate application found */
                    nRetVal = 1;
                }
            }
        }
    }
}

```

```
        }  
    }  
  
    /* return to caller */  
    return nRetVal;  
}
```

It is further noted that codec 7-303 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 7-210. For example, codec 7-303, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 7-210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data pathways that are commonly used for recording, such that the utilized data pathway is governed by the copyright compliance mechanism 7-300. In one embodiment, this can be performed within a driver shim, e.g., wave driver shim 7-309 of Figures 7-5A and 7-5B.

A driver shim can be utilized for nearly any software output device, e.g., a standard Windows™ waveform output device, e.g., Windows™ Media Player, or hardware output device, e.g., speakers or headphones. Client computer system 7-210 is configured such that the driver shim (e.g., 7-309 of Figures 7-5A, 7-5B, 7-5C, and 7-5D) will appear as the default waveform media device to client level application programs. Thus, requests for processing of waveform media input and/or output will pass through the driver shim prior to being forwarded to the actual waveform audio driver, media device driver 7-505 of Figures 7-5A and 7-5B. Such waveform input/output suppression can be triggered by other components of CCM 7-300, e.g., agent 7-304, to be active when a recording operation is initiated by a client computer system, e.g., 7-210, during the play back of media files which are subject to the DMCA.

It is noted that alternative driver shims can be implemented for nearly any waveform output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices, video, graphic and/or alphanumeric media files and video input and output devices.

The following C++ source code is an exemplary implementation of a portion of a codec 7-303 and/or a custom media device driver 7-307 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```

DWORD
_stdcall
widMessage(UINT          uDevId,
            UINT          uMsg,
            DWORD         dwUser,
            DWORD         dwParam1,
            DWORD         dwParam2)
{
    BOOL      bSkip;          /* flag indicating operation to be skipped */
    HWND      hWndMon;        /* handle to main window for monitor */
    DWORD     dwRetVal;        /* return value for operation */

    /* initialize variables */
    bSkip = FALSE;
    dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;

    if (uMsg == WIDM_START)
    {
        /* attempt to locate window for monitor application */
        if ( (hWndMon = FindMonitorWindow())
            != (HWND) 0)
        {
            /* obtain setting for driver */
            bDrvEnabled = ( SendMsg(hWndMon,
                                   uiRegMsg,
                                   0,

```

```

                                0)
                                = 0)
                                ? FALSE : TRUE;
        }

        if (bDrvEnabled == TRUE)
        {
            /* indicate error in operation */
            dwRetVal = MMSYSERR_NOMEM;

            /* indicate operation to be skipped */
            bSkip = TRUE;
        }
    }

    if (bSkip == FALSE)
    {
        /* invoke entry point for original driver */
        dwRetVal = CallWidMessage(uDevId, uMsg, dwUser, dwParam1,
dwParam2);
    }

    /* return to caller */
    return dwRetVal;
}

```

It is noted that when properly configured, system hook 7-305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 7-210 to 7-230. In one embodiment, system hook 7-305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a Microsoft Windows™ operating environment, and accordingly can be implemented in a variety of ways. However, it is also noted that system hook 7-305 can be readily adapted for implementation in alternative operating systems, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 7-3, copyright compliance mechanism 7-300 also includes one or more skins 7-306, which can be designed to be installed in a client computer system, e.g., 7-210 to 7-230. In one embodiment, skins 7-306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 7-306 are customizable interfaces that, in one embodiment, are displayed on a display device (e.g., 7-105) of computer system 7-210 and provide functionalities for user interaction of delivered media content. Additionally, skins 7-306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

Furthermore, when system hook 7-305 is unable to govern a function of the media player application operable on a client computer system, e.g., 7-210, such that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 7-306 can be implemented to provide compliance.

Differing skins 7-306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 7-306a may be configured for utilization with a media content file protected under a non-interactive agreement (DMCA), such that skin 7-306a may not include a pause function, a stop function, a selector function, and/or a save function, etc. Another skin, e.g., skin 7-306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with “no save” agreement (DMCA), such that skin 7-306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with “save” agreement, a save or a burn to CD function.

Still referring to Figure 7-3, it is further noted that in the present embodiment, each skin 7-306 can have a unique name and signature. In one embodiment, skin 7-306 can be implemented, in part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 7-306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD5 hash, without knowledge of the MD5 hash table, changing the name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance mechanism 7-300 verifies skin 7-306, MD5 hash tables advantageously provide a deterrent against modifications made to the skin.

In one embodiment, copyright compliance mechanism 7-300 also includes one or more custom media device driver(s) 7-307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 7-210, can be configured to utilize a custom media device application, e.g., custom media device 7-310 (shown in Figure 7-5B, 7-5C, and 7-5D), to control unauthorized recording of media content files. A custom media device application can be, but is not limited to, a custom media audio device application for media files having sound content, a custom video device application for media files having graphical and/or alphanumeric content, etc. In one embodiment, custom media device 7-310 of Figure 7-5B is an emulation of the custom media device driver 7-307. With reference to audio media, the emulation is performed in a waveform audio driver associated with custom media device 7-310. Driver 7-307 is configured to receive a media file being outputted by system 7-210 prior to the media file being sent to a media output device, e.g., media output device 7-570, and/or a media

output application, e.g., recording application 7-502. Examples of a media output device includes, but is not limited to, a video card for video files, a sound card for audio files, etc. Examples of a recording application can include, but is not limited to, CD burner applications for writing to another CDs, ripper applications which capture the media file and change the format of the media file, e.g., from a MP3 file to a .wav file. In one embodiment, client computer system 7-210 is configured with a custom media device driver 7-307 emulating custom media device 7-310, and which is system 7-210's default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of skin 7-306 or a custom web based player application or as part of a CCM 7-300 installation bundle, for forcing or requiring system 7-210 to have driver 7-307 as the default driver.

Therefore, when a media content file is received by system 7-210 from server 7-251, the media content file is playable, provided the media content file passes through the custom media device application (e.g., 7-310 of Figure 7-5B), emulated by custom media device driver 7-307, prior to being outputted. However, if an alternative media player application is selected, delivered media files from server 7-251 will not play on system 7-210.

Thus, secured media player applications would issue a media request to the driver, e.g., 7-307, for the custom media device 7-310 which then performs necessary media input suppression, e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 7-307 to a Windows™ waveform audio driver operable on system 7-210,

thus reducing instances of incompatibilities with existing media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 7-300, e.g., agents 7-304, system hooks 7-305, and skins 7-306, or a combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of ways including, but limited to, those found at developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 7-210 being configured with a custom media device as the default device driver e.g., device 7-310 of Figures 7-5B, 7-5C, and 7-5D, emulated by a custom media device driver 7-307, those media player applications that require their particular device driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured music. Further advantageous is that an emulated custom media device provides no native support for those media player applications used as a recording mechanism, e.g., DirectSound capture, (direct sound 7-504 of Figures 7-5A, 7-5B, 7-5C, and 7-5D) etc., that are able to bypass user-mode drivers for most media devices. Additionally, by virtue of the media content being sent through device driver 7-307, thus effectively disabling unauthorized saving/recording of media files, in one embodiment, media files that are delivered in a secured delivery system do not have to be encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-encrypted media files utilizing less storage space and network resources than encrypted media files, networks having limited resources can utilize the functionalities of driver 7-307 of CCM 7-300 to provide compliance

with copyright restrictions and/or licensing agreements applicable with a media content file without having the processing overhead of encrypted media files.

Figure 7-4 is an illustration of an exemplary system 7-400 for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention. Specifically, system 7-400 illustrates web server 7-250, content server 7-251, or a combination of web server 7-250 and content server 7-251 installing a copyright compliance mechanism (e.g., 7-300) in a client's computer system (e.g., 7-210) for controlling media file distribution and controlling user access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 7-210 can communicatively couple with a network (e.g., 7-200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 7-250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 7-250 can initiate a registration process with the requesting client 7-210. Client registration can be accomplished in a variety of ways. For example, web server 7-250 may deliver to a client 7-210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 7-250 including, but not limited to, user's name, address, phone number, credit card number, verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 7-4, web server 7-250 can, in one embodiment, detect information related to the client's computer system, e.g., 7-210, and store that information in a user/media database 7-450. For example, web server 7-250 can detect a unique identifier of client computer system 7-210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network interface card) of client computer system 7-210 or the MAC address of the network interface adapter integrated on the motherboard of system 7-210. It is understood that a NIC enables a client computer system 7-210 to access web server 7-250 via Internet 7-201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 7-250 can, in one embodiment, detect and store (also in database 7-450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 7-210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 7-450) within web server 7-250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 7-210, all or part of this information can be installed in client computer system 7-210. In one embodiment, client computer system 7-210 information can be in the form of a cookie. Web server 7-250 then verifies that the user and client computer system 7-210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 7-250 installs a copyright compliance mechanism (e.g., 7-300) into the client's computer system, e.g., 7-210, in one embodiment of the present invention. It is noted that web server 7-250 may not initiate

installation of CCM 7-300 until the user ID, password, and client computer system 7-210 information is verified. A variety of common techniques can be employed to install an entire CCM 7-300, portions of components, entire components, and/or combinations or a function of components. For example, copyright compliance mechanism 7-300 can be installed in a hidden directory within client computer system 7-210, thereby preventing unauthorized access to it. In one embodiment of the present invention, it is noted that unless CCM 7-300 is installed in client computer system 7-210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 7-250 and/or content server 7-251,

Referring still to Figure 7-4, upon completion of client registration and installation of CCM 7-300, client computer system 7-210 can then request a media play list or a plurality of play lists, etc. In response, web server 7-250 determines whether the user of client computer system 7-210 is authorized to receive the media play list associated with the request. In one embodiment, web server 7-250 can request the username and password. Alternatively, web server 7-250 can utilize user database 7-450 to verify that computer 7-210 is authorized to receive a media play list. If client computer 7-210 is not authorized, web server 7-250 can initiate client registration, as described herein. Additionally, web server 7-250 can disconnect computer 7-210 or redirect it to an alternative web site. Regardless, if the user and client computer system 7-210 are not authorized, web server 7-250 will not provide the requested play list to client computer system 7-210.

However, if client computer system 7-210 is authorized, web server 7-210 can check copyright compliance mechanism 7-300 within data base 7-450 to determine if it, or any of the components therein, have been updated since the last time client computer system 7-210 logged in to web server 7-250. If a component of CCM 7-300 has been updated, web server

7-250 can install the updated component and/or a more current version of CCM 7-300 into client computer system 7-210, e.g., via Internet 7-201. If CCM 7-300 has not been updated, web server 7-250 can then deliver the requested media play list to system 7-210 via Internet 7-201 along with an appended user key or user identification (ID). It is noted that user database 7-450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 7-210. Subsequently, the user of client computer system 7-210 can utilize the received media play list in combination with the media player application operating on system 7-210 to transmit a delivery request for one or more desired pieces of media content from web server 7-250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 7-4, upon receiving the media content delivery request, web server 7-250 can then check the validity of the requesting media application and the attached user key. In one embodiment, web server 7-250 can utilize user database 7-450 to check their validity. If either or both are invalid, web server 7-250, in one embodiment, can redirect unauthorized client computer system 7-210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 7-300 verifies that skins 7-306 are installed in client computer system 7-210. Additionally, CCM 7-300 further verifies that system hook(s) 7-305 have been run or are running to govern certain functions of those media player applications operable within client computer system 7-210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 7-300 further diverts and/or redirects certain pathways that are commonly used for recording, e.g., driver 7-307 of Figure 7-5A, device 7-310 of Figure 7-5B, device 7-570 of Figure 7-5C, and driver 7-505 of Figure 7-5D. Once CCM 7-300 has performed the above described functions, web server 7-250 then, in one embodiment, issues

to the client computer 7-210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 7-210 receiving the redirect command from web server 7-250, the media player application operating on client computer system 7-210 automatically transmits a new request and the time sensitive access key to content server 7-251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 7-251. If invalid, unauthorized client computer 7-210 is redirected by content server 7-250 to protect against abuse of the system and unauthorized access to content server 7-251. If the time sensitive access key is valid, content server 7-251 retrieves the desired media content from content database 7-451 and delivers it to client computer system 7-210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 7-210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 7-210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 7-4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g., title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 7-300 that identifies the media file as

originating from a content server 7-251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 7-300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or the entire CCM 7-300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to modification of copyright compliance mechanism 7-300, modified portions of, or the entire updated CCM 7-300 can easily be installed in client computer system 7-210 in a variety of ways. For example, the updated CCM 7-300 can be installed during client interaction with web server 7-250, during user log-in, and/or while client computer system 7-210 is receiving the keyed play list.

Referring still to Figure 7-4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 7-251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be stored in a database 7-450, content database 7-451 or in some combination of databases 7-450 and 7-451. It is further noted that the messages being passed back and forth between client computer system 7-210 and web server 7-250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting

with adds, public domain encryption programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be similarly encrypted. Furthermore, in another embodiment, the media files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 7-307.

Subsequent to media file decryption, the media file may be passed through CCM 7-300, e.g., a coder/decoder 7-303, to a media player application operating on client computer system 7-210, e.g. playback application 7-501 of Figures 7-5A, 7-5B, 7-5C, 7-5D, and 7-6A, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 7-306 of Figure 7-3. A skin 7-306 may be necessary when CCM 7-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client computer system 7-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 7-303 will repeatedly ensure that

CCM 7-300 rules are being enforced at any particular moment during media playback, shown as step 7-650 of Figure 7-6C.

As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 7-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 7-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These presentation stoppages are collectively shown as step 7-651 of Figure 7-6C.

If the rules, in accordance with CCM 7-300, are enforced, the codec/decoder 7-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 7-210. The newly retrieved portion of the media file is then presented by the client's media player application. While the newly retrieved portion is presented, CCM 7-300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 7-300 can detect undesired activities and enforces those rules as defined by CCM 7-300.

Figure 7-5A is an exemplary logic/bit path block diagram 7-500A showing utilization of a wave shim driver, e.g., wave shim driver 7-309 of Figure 7-3, in conjunction with copyright compliance mechanism 7-300, for selectively controlling recording of copyrighted media received by a client computer system, e.g., system 7-210, in one embodiment of the present invention. Copyright compliance mechanism 7-300 is, in one embodiment, installed and operational on client system 7-210 in the manner described herein.

In one embodiment, a copyright compliance mechanism 7-300 is shown as being communicatively coupled with a media playback application 7-501 via connection 7-520. Therefore, CCM 7-300 is enabled to communicate with playback application 7-501. In one embodiment, CCM 7-300 can be integrated into a media playback application. CCM 7-300 is also coupled to and controls a selectable switch 7-311 in wave shim driver 7-309 (as described in Figure 7-3) via connection 7-522. CCM 7-300 is further coupled to and controls a selectable switch 7-511 in direct sound 7-504 via connection 7-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., 7-499, CCM 7-300 controls whether switches 7-311 and 7-511 are open (shown), thus preventing incoming media 7-499 from reaching a media recording application, or closed (not shown) to allow recording of incoming media 7-499.

For example, incoming media 7-499 may originate from a content server, e.g., 7-251, coupled to system 7-210. In another example, incoming media 7-499 may originate from a personal recording/electronic device, e.g., a MP3 player/recorder or similar device, coupled to system 7-210. Alternatively, incoming media 7-499 may originate from a magnetic, optical or alternative media storage device inserted into a media device player coupled to

system 7-210, e.g., a CD or DVD inserted into a CD or DVD player, a hard disk in a hot swappable hard drive, an SD (secure digital card) inserted into a SD reader, and the like. In yet another example, incoming media 7-499 may originate from another media player application or media recording application. Incoming media 7-499 may also originate from a satellite radio feed (e.g., XM radio), a personal communication device (e.g., a mobile phone), a cable television radio input (e.g., DMX (digital music express)), or a set-top box. It is noted that incoming media 7-499 can originate from nearly any source that can be coupled to system 7-210. However, regardless of the source of incoming media 7-499, embodiments of the present invention, described herein, can prevent unauthorized recording of the media.

Figure 7-5A shows a media playback application 7-501, e.g., an audio, video, or other media player application, operable within system 7-210 and configured to receive incoming media 7-499. Playback application 7-501 can be a playback application provided by an operating system, e.g., Media Player for Windows™ by Microsoft, a freely distributed playback application downloadable from the Internet, e.g., RealPlayer or LiquidAudio, a playback application provided by a webcaster, e.g., PressPlay, or a playback application commercially available.

Figure 7-5A shows media device driver 7-505 which, in one implementation, may be a software driver for a sound card coupled to system 7-210 having a media output device 7-570, e.g., speakers or headphones, coupled therewith for media files having audio content. In another implementation, media device driver 7-505 may be a software driver for a video card coupled with a display device, e.g., 7-105, for displaying media files having alphanumeric and/or graphical content, and so on. With reference to audio files, it is well known that a majority of recording applications assume a computer system, e.g., 7-210, has a sound card.

disposed therein, providing full-duplex sound functionality to system 7-210. This means media output driver 7-505 can simultaneously cause playback and recording of incoming media files 7-499. For example, media device driver 7-505 can playback media 7-499 along wave-out line 7-539 to media output device 7-570 (e.g., speakers for audible playback) via wave-out line 7-580 while outputting media 7-499 on wave-out line 7-540 to eventually reach recording application 7-502.

For purposes of Figures 7-5A, 7-5B, 7-5C, and 7-5D, the terms wave-in line and wave-out line are referenced from the perspective of media device driver 7-505. Additionally, for the most part, wave-in lines are downwardly depicted and wave-out lines are upwardly depicted in Figures 7-5A, 7-5B, 7-5C, and 7-5D.

Continuing with Figure 7-5A, playback application 7-501 is coupled with an operating system (O/S) multimedia subsystem 7-503 and direct sound 7-504 via wave-in lines 7-531 and 7-551 respectively. O/S multimedia subsystem 7-503 is coupled to a wave shim driver 7-309 via wave-in line 7-533 and wave-out line 7-546. O/S multimedia subsystem 7-503 is also coupled to a recording application 7-502 via wave-out line 7-548. Operating system (O/S) multimedia subsystem 7-503 can be any O/S multimedia subsystem, e.g., a Windows™ multimedia subsystem for system 7-210 operating under a Microsoft O/S, a QuickTime™ multimedia subsystem for system 7-210 operating under an Apple O/S, and so on. Playback application 7-501 is also coupled with direct sound 7-504 via wave-in line 7-551.

Direct sound 7-504, in one instance, may represent access to a hardware acceleration feature in a standard audio device, enabling lower level access to components within media

device driver 7-505. In another instance, direct sound 7-504 may represent a path that can be used by a recording application, e.g., Total Recorder, that can be further configured to bypass the default device driver, e.g., media device driver 7-505 to capture incoming media 7-499 for recording. For example, direct sound 7-504 can be enabled to capture incoming media 7-499 via wave-in line 7-551 and unlawfully output media 7-499 to a recording application 7-502 via wave-out line 7-568, as well as media 7-499 eventually going to media device driver 7-505, the standard default driver.

Still referring to Figure 7-5A, wave shim driver 7-309 is coupled with media device driver 7-505 via wave-in line 7-537 and wave-out line 7-542. Media device driver 7-505 is coupled with direct sound 7-504 via wave-in line 7-553 which is shown to converge with wave-in line 7-537 at media device driver 7-505. Media device driver 7-505 is also coupled with direct sound 7-504 via wave-out line 7-566.

Wave-out lines 7-542 and 7-566 are shown to diverge from wave-out line 7-540 at media device driver 7-505 into separate paths. Wave-out line 7-542 feeds into wave shim driver 7-309 and wave-out line 7-566 feeds into direct sound 7-504. When selectable switch 7-311 and 7-511 are open (shown), incoming media 7-499 cannot flow to recording application 7-502, thus preventing unauthorized recording of it.

For example, incoming media 7-499 is received at playback application 7-501. Playback application 7-501 activates and communicates to CCM 7-300 regarding copyright restrictions and/or licensing agreements applicable to incoming media 7-499. If recording restrictions apply to media 7-499, CCM 7-300 can, in one embodiment, open switches 7-311 and 7-511, thereby blocking access to recording application 7-502, effectively preventing

unauthorized recording of media 7-499. In one embodiment, CCM 7-300 can detect if system 7-210 is configured with direct sound 7-504 selected as the default driver to capture incoming media 7-499, via wave-in line 7-551, or a recording application is detected and/or a hardware accelerator is active, such that wave driver shim 7-309 can be bypassed by direct sound 7-504. Upon detection, CCM 7-300 can control switch 7-511 such that the output path, wave-out line 7-568, to recording application 7-502 is blocked. It is further noted that CCM 7-300 can detect media recording applications and devices as described herein, with reference to Figure 7-3.

Alternatively, if media device driver 7-505 is selected as the default driver, incoming media 7-499 is output from playback application 7-501 to O/S multimedia subsystem 7-503 on wave-in line 7-531. From subsystem 7-503, media 7-499 is output to wave shim driver 7-309 via wave-in line 7-533. The wave shim driver 7-309 was described herein with reference to Figure 7-3. Media 7-499 is output from wave shim driver 7-309 to media device driver 7-505 via wave-in line 7-537. Once received by media device driver 7-505, media 7-499 can be output via wave-out line 7-539 to a media output device 7-570 coupled therewith via wave-out line 7-580. Additionally, media device driver 7-505 can simultaneously output media 7-499 on wave-out line 7-540 back to wave shim driver 7-309. Dependent upon recording restrictions applicable to media 7-499, CCM 7-300 can, in one embodiment, close switch 7-311 (not shown as closed), thereby allowing media 7-499 to be output from wave shim driver 7-309 to subsystem 7-503 (via wave-out line 7-546) and then to recording application 7-502 via wave-out line 7-548. Alternatively, CCM 7-300 can also open switch 7-311, thereby preventing media 7-499 from reaching recording application 7-502.

It is particularly noted that by virtue of CCM 7-300 controlling both switches 7-311 and 7-511, and therefore controlling wave-out line 7-548 and wave-out line 7-568 leading into recording application 7-502, incoming media files, e.g., media 7-499, can be prevented from being recorded in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media. It is also noted that embodiments of the present invention in no way interfere with or inhibit the playback of incoming media 7-499.

Figure 7-5B is an exemplary logic/bit path block diagram 7-500B of a client computer system, e.g., 7-210, configured with a copyright compliance mechanism 7-300 for preventing unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 7-300 is, in one embodiment, coupled with and operational on client system 7-210 in the manner with reference to Figures 7-4, 7-5A, 7-5C, 7-5D, 7-6, and 7-7.

Diagram 7-500B of Figure 7-5B is similar to diagram 7-500A of Figure 7-5A, with a few changes. Particularly, diagram 7-500B includes a custom media device 7-310 communicatively interposed between and coupled to O/S multimedia subsystem 7-503 and wave shim driver 7-309. Custom media device 7-310 is coupled to O/S multimedia subsystem via wave-in line 7-533 and wave-out line 7-546. Custom media device 7-310 is coupled with wave shim driver 7-309 via wave-in line 7-535 and wave-out line 7-544. Additionally, custom media device 7-310 is coupled with direct sound 7-504 via wave-in line 7-553 which converges with wave-in line 7-533 and wave-out line 7-566 which diverges from wave-out line 7-546, in one embodiment.

Also added to Figure 7-5B is a media hardware output device 7-570 that is coupled to media device hardware driver 7-505 via line 7-580. Media hardware output device 7-570 can be, but is not limited to, a sound card for audio playback, a video card for video, graphical, alphanumeric, etc, output, and the like.

In one embodiment, CCM 7-300 is communicatively coupled with playback application 7-501 via connection 7-520, waveform driver shim 7-309 via connection 7-522, and custom media device 7-310, via connection 7-521. CCM 7-300 is coupled to and controls a selectable switch 7-311 in waveform driver shim 7-309 via connection 7-522. CCM 7-300 is also coupled to and controls a selectable switch 7-312 in custom audio device 7-310 via connection 7-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 7-499, CCM 7-300 controls whether switches 7-311 and 7-312 are open (shown), thus preventing the incoming media 7-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 7-499.

Continuing with Figure 7-5B, direct sound 7-504 is shown coupled with custom media device 7-310 via wave-in line 7-553, instead of being coupled with media device driver 7-505 (Figure 7-5A). In one embodiment, custom audio device 7-310 mandates explicit selection through system 7-210, meaning that custom audio device 7-310 needs to be selected as a default driver of system 7-210. By virtue of having the selection of custom media device 7-310 as the default driver of system 7-210, the data path necessary for direct sound 7-504 to capture the media content is selectively closed.

For example, incoming media 7-499 originating from nearly any source with reference to Figure 7-5A is received by media playback application 7-501 of system 7-210. Playback application 7-501 communicates to CCM 7-300, via connection 7-520, to determine whether incoming media 7-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 7-501 communicates with CCM 7-300 to control switch 7-311 and 7-312 accordingly. In the present example, recording of incoming media 7-499 would violate applicable restrictions and/or agreements and therefore switch 7-312 is in an open position, such that the output path to recording application 7-502, e.g., wave-out line 7-548 and/or wave-out line 7-568, is effectively blocked, thereby preventing unauthorized recording of media 7-499.

Alternatively, if media device driver 7-505 is selected as the default driver, incoming media 7-499 continues from O/S multimedia subsystem 7-503, through custom audio device 7-310, wave driver shim 7-309, and into media device driver 7-505 where media 7-499 can be simultaneously output to media output device 7-570 via line 7-580, and output on wave-out line 7-540 to wave-and outputted by media device driver 7-505 to wave shim driver 7-309 on wave-out line 7-542. However, by virtue of CCM 7-300 controlling switch 7-311, wave-out line 7-544 which eventually leads to recording application 7-502 is blocked, thus effectively preventing unauthorized recording of media 7-499.

It is particularly noted that by virtue of CCM 7-300 controlling both switches 7-311 and 7-312 and therefore controlling wave-out line 7-548 and wave-out line 7-568, any incoming media files, e.g., incoming media 7-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 7-5B, it is further noted that custom media device 7-310 allows for unfettered playback of incoming media 7-499. Additionally, at any time during playback of media 7-499, custom media device 7-310 can be dynamically activated by CCM 7-300.

Figure 7-5C is an exemplary logic/bit path block diagram 7-500C of a client computer system, e.g., 7-210, configured with a copyright compliance mechanism 7-300 for preventing unauthorized output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 7-300 is, in one embodiment, coupled with and operational on client system 7-210 in the manner with reference to Figures 7-4, 7-5A, 7-5B, 7-5D, 7-6, and 7-7.

Diagram 7-500C of Figure 7-5C is similar to diagram 7-500B of Figure 7-5B, with a few changes. Particularly, diagram 7-500C includes a media hardware output device 7-570 that is coupled with a media device driver 7-505. In one embodiment, media hardware output device 7-570 can be a S/PDIF (Sony/Phillips Digital Interface) card for providing multiple outputs, e.g., an analog output 7-573 and a digital output 7-575. An alternative media hardware output device providing similar digital output can also be implemented as device 7-570 including, but not limited to, a USB (universal serial bus) output device and/or an externally accessible USB port located on system 7-210, a FireWire (IEEE1394) output device and/or an externally accessible FireWire port located on system 7-210, with wireline or wireless functionality. In the present embodiment, media hardware output device 7-570 is shown to include a switch 7-571 controlled by CCM 7-300 via communication line 7-523, similar to switches 7-311 and 7-312, for controlling output of incoming media 7-499.

In one embodiment, CCM 7-300 is communicatively coupled with playback application 7-501 via connection 7-520, waveform driver shim 7-309 via connection 7-522, custom media device 7-310, via connection 7-521, and media hardware output device 7-570 via connection 7-523. CCM 7-300 is coupled to and controls a selectable switch 7-311 in waveform driver shim 7-309 via connection 7-522. CCM 7-300 is also coupled to and controls a selectable switch 7-312 in custom audio device 7-310 via connection 7-521. CCM 7-300 is further coupled to and controls a selectable switch 7-571 in media hardware output device 7-570 via connection 7-523. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 7-499, CCM 7-300 controls whether switches 7-311 and 7-312 are open (shown), thus preventing the incoming media 7-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 7-499. Additionally, CCM 7-300 controls whether switch 7-571 is open (shown), thus preventing incoming media 7-499 from being output from digital output 7-575 of media hardware output device 7-570, or closed (not shown) to allow incoming media 7-499 to be output from media hardware output device 7-570.

By controlling media hardware output device 7-570, copyright compliance mechanism 7-300 can prevent unauthorized output of incoming media 7-499 to, e.g., a digital recording device that may be coupled with digital output 7-575 of media hardware output device 7-570. Accordingly, in one embodiment, CCM 7-300 is enabled to also detect digital recording devices that may be coupled to a digital output line, e.g., 7-571, of a media hardware output device, e.g., 7-570. Examples of a digital recording device that can be coupled to media hardware output device 7-570 can include, but is not limited to, mini-disc recorders, MP3 recorders, personal digital recorders, digital recording devices coupled with

multimedia systems, personal communication devices, set-top boxes, and/or nearly any digital device that can capture an incoming media 7-499 being output from a media hardware output device 7-570, e.g. a sound card.

Continuing with Figure 7-5C, direct sound 7-504 is shown coupled with custom media device 7-310 via wave-in line 7-553, instead of being coupled with media device driver 7-505 (Figure 7-5A). In one embodiment, custom audio device 7-310 mandates explicit selection through system 7-210, meaning that custom audio device 7-310 is needs to be selected as a default driver of system 7-210. By virtue of having the selection of custom media device 7-310 as the default driver of system 7-210, the data path necessary for direct sound 7-504 to capture the media content is selectively closed.

For example, incoming media 7-499 originating from nearly any source with reference to Figure 7-5A is received by media playback application 7-501 of system 7-210. Playback application 7-501 communicates to CCM 7-300, via connection 7-520, to determine whether incoming media 7-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 7-501 communicates with CCM 7-300 to control switch 7-311, 7-312, and 7-571 accordingly. In the present example, recording of incoming media 7-499 would violate applicable restrictions and/or agreements and therefore switch 7-312 is in an open position, such that the output path to recording application 7-502, e.g., wave-out line 7-548 and/or wave-out line 7-568, is effectively blocked, thereby preventing unauthorized recording of media 7-499.

Alternatively, if media device driver 7-505 is selected as the default driver, incoming media 7-499 continues from O/S multimedia subsystem 7-503, through custom audio device

7-310, wave driver shim 7-309, and into media device driver 7-505 where media 7-499 can be simultaneously output to media output device 7-570 via line 7-580, and output on wave-out line 7-540 to wave-and outputted by media device driver 7-505 to wave shim driver 7-309 on wave-out line 7-542. However, by virtue of CCM 7-300 controlling switch 7-311, wave-out line 7-544 which eventually leads to recording application 7-502 is blocked, thus effectively preventing unauthorized recording of media 7-499.

It is particularly noted that by virtue of CCM 7-300 controlling both switches 7-311 and 7-312 and therefore controlling wave-out line 7-548 and wave-out line 7-568, any incoming media files, e.g., incoming media 7-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 7-5C, it is particularly noted that although CCM 7-300 can prevent unauthorized recording of incoming media 7-499 by controlling switches 7-311 and 7-312, thus preventing incoming media 7-499 from reaching recording application 7-502, controlling switches 7-311 and 7-312 do nothing to prevent incoming media 7-499 from being captured by a peripheral digital device, e.g., a mini-disc recorder, etc., coupled to a digital output 7-575 of device 7-570. Thus, by also controlling the output, via digital output 7-575 of media hardware output device 7-570, through control of switch 7-571, CCM 7-300 can prevent unauthorized capturing of incoming media 7-499 during output, e.g., on a sound card for audio files, a video card for video and/or graphical files, regardless of whether incoming media 7-499 is received in a secure and encrypted manner. However, when switch 7-571 is in a closed position, incoming media 7-499 may be played back in an unfettered manner. Additionally, at any time during playback of media 7-499, switch 7-312 of custom

media device 7-310, switch 7-311 of media device driver 7-309, and/or switch 7-571 of media hardware output device 7-570 can be dynamically activated by CCM 7-300.

Figure 7-5D is an exemplary logic/bit path block diagram 7-500D of a client computer system, e.g., 7-210, configured with a copyright compliance mechanism 7-300 for preventing unauthorized kernel based output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 7-300 is, in one embodiment, coupled with and operational on client system 7-210 in the manner described herein with reference to Figures 7-4, 7-5A, 7-5B, 7-5C, 7-6, and 7-7.

Diagram 7-500D of Figure 7-5D is similar to diagram 7-500C of Figure 7-5C, with some changes. Particularly, diagram 7-500D includes a kernel streaming mechanism 7-515, e.g., DirectKS, that is coupled with a media device driver 7-505. In one embodiment, DirectKS 7-515 can be used for establishing a direct connection with media device driver 7-505. In the present embodiment, media device driver 7-505 is shown to include a switch 7-511 controlled by CCM 7-300 via communication line 7-524, that is similar to switches 7-311, 7-312, and 7-571, for controlling output of incoming media 7-499.

In one embodiment, CCM 7-300 is communicatively coupled with: playback application 7-501 via connection 7-520, waveform driver shim 7-309 via connection 7-522, custom media device 7-310, via connection 7-521, and media device driver 7-505 via connection 7-524. Specifically, CCM 7-300 is coupled to and controls a selectable switch 7-311 in waveform driver shim 7-309 via connection 7-522. CCM 7-300 is also coupled to and controls a selectable switch 7-312 in custom audio device 7-310 via connection 7-521. CCM

7-300 is further coupled to and controls a selectable switch 7-511 in media device driver 7-505 via connection 7-524. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 7-499, CCM 7-300 controls whether switches 7-311 and 7-312 are open (shown), thus preventing the incoming media 7-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 7-499. Additionally, CCM 7-300 controls whether switch 7-511 is open (shown), thus preventing incoming media 7-499 from being returned from media device driver 7-505 to playback application 7-501, where DirectKS 7-515 can capture incoming media 7-499 and redirect it to recording application 7-502 to create an unauthorized copy or recording of incoming media 7-499. CCM 7-300 can also control whether switch 7-511 is closed (not shown) to allow incoming media 7-499 to be returned to playback application 7-501, where DirectKS 7-515 can capture and redirect incoming media 7-499 to recording application 7-502.

DirectKS 7-515, in one embodiment, may represent a kernel streaming mechanism that is adapted to establish a direct connection with a media device driver 7-505 of an operating system operable on client computer system 7-210, enabling kernel level access to media device driver 7-505. A kernel streaming mechanism can be implemented for the purpose of precluding utilization of standard audio APIs (application programming interfaces) to play or record media content, with particular attention paid to those playback applications with low latency requirements. DirectKS 7-515 can bypass existing APIs and communicate with media device driver 7-505. DirectKS 7-515 can be readily adapted to work in conjunction with a playback application, e.g., 7-501, to capture and redirect incoming media 7-499 to recording application 7-502, via wave-out line 7-588. Accordingly, DirectKS 7-515 can be implemented to create unauthorized media recordings.

By controlling media device driver 7-505, copyright compliance mechanism 7-300 can prevent unauthorized output of incoming media 7-499 to, e.g., a digital recording device 7-529 that may be coupled with recording application 7-502. In one embodiment, media device driver 7-505 is configured through the kernel mixer (not shown) to control the data path. Additionally, in one embodiment, CCM 7-300 is enabled to also detect a kernel streaming mechanism 7-515 (e.g., DirectKS) that may be operable on client computer system 7-210, as described herein with reference to Figure 7-3.

In one embodiment, custom media device 7-310 mandates explicit selection through system 7-210, meaning that custom media device 7-310 is needs to be selected as a default driver of system 7-210. By virtue of having the selection of custom media device 7-310 as the default driver of system 7-210, the data path necessary for direct sound 7-504 to capture the media content is selectively closed.

For example, incoming media 7-499 originating from nearly any source with reference to Figure 7-5A is received by media playback application 7-501 of system 7-210. Playback application 7-501 communicates to CCM 7-300, via connection 7-520, to determine whether incoming media 7-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 7-501 communicates with CCM 7-300 to control switches 7-311, 7-312, 7-571, and 7-511, accordingly. In the present example, recording of incoming media 7-499 would violate applicable restrictions and/or agreements and therefore switch 7-511 is in an open position, such that the output path to recording application 7-502, e.g., wave-out line 7-548 and/or wave-out line 7-568 and/or wave-out line 7-588, is effectively blocked, thereby preventing unauthorized recording of media 7-499.

Still referring to Figure 7-5D, it is particularly noted that although CCM 7-300 can prevent unauthorized recording of incoming media 7-499 by controlling switches 7-311, 7-312, and 7-571, thus preventing incoming media 7-499 from reaching recording application 7-502, controlling switches 7-311, 7-312, and 7-571, do nothing to prevent incoming media 7-499 from being returned to recording application 7-502 by a kernel streaming mechanism 7-515(e.g., DirectKS), which enables capturing and redirecting of incoming media 7-499 to recording application 7-502, via wave-out line 7-588. Thus, by also controlling switch 7-511 of media device driver 7-505, CCM 7-300 can prevent kernel streaming mechanism 7-515 from returning incoming media 7-499 to recording application 7-502, thereby preventing incoming media 7-499 from being captured and redirected to recording application 7-502 in an attempt to create and unauthorized copy and/or recording of incoming media 7-499. However, when switch 7-511 is in a closed position, incoming media 7-499 may be returned to a recording application 7-502, such that recording could be possible, provided recording does not violate copyright restrictions applicable to incoming media 7-499. Additionally, at any time during playback of media 7-499, switch 7-312 of custom media device 7-310, switch 7-311 of wave shim driver 7-309, and/or switch 7-511 of media device driver 7-505 can be dynamically activated by CCM 7-300.

Figure 7-6A is an block diagram of a media file, e.g., incoming media 7-499, adapted to be received by a playback application, e.g., 7-501 of Figures 7-5A, 7-5B, 7-5C, and 7-5D, configured with an indicator 7-605 for enabling incoming media 7-499 to comply with rules according to the SCMS (serial copy management system). When applicable to a media file, e.g., 7-499, the SCMS allows for one copy of a copyrighted media file to be made, but not for copies of copies to be made. Thus, if incoming media 7-499 can be captured by a recording

application, e.g., 7-502 of Figures 7-5A, 7-5B, 7-5C, and/or 7-5D, and/or a recording device, e.g. 7-529, and/or a peripheral recording device and/or a recording application coupled to a digital output of a media hardware output device, e.g., digital output 7-575 of media hardware output device 7-570 of Figures 7-5B, 7-5C, and 7-5D, and/or a kernel streaming mechanism 7-515, e.g., DirectKS of Figure 7-5D, unauthorized copying and/or recording may be accomplished.

Playback application 7-501 is coupled with CCM 7-300 via communication line 7-520 in a manner analogous to Figures 7-5A, 7-5B, 7-5C, and/or 7-5D. Although not shown in Figure 7-6, it is noted that CCM 7-300 is also coupled to switches 7-311 and 7-511 as shown in Figure 7-5A, switches 7-311 and 7-312 in Figure 7-5B, switches 7-311, 7-312, and 7-571 in Figure 7-5C, and switches 7-312, 7-311, 7-571, and 7-511, in Figure 7-5D.

In one embodiment, an indicator 7-605 is attached to incoming media 7-499 for preventing unauthorized copying or recording in accordance with the SCMS. In one embodiment, indicator 7-605 can be a bit that may be transmitted prior to beginning the delivery of incoming media 7-499 to playback application 7-501. In another embodiment, indicator 7-605 may be placed at the beginning of the bit stream of incoming media 7-499. In another embodiment, indicator 7-605 may be placed within a frame period of incoming media 7-499, e.g., every fifth frame, or any other desired frame period. In another embodiment, indicator 7-605 may be transmitted at a particular time interval or intervals during delivery of the media file, e.g. incoming media 7-499. Thus, indicator 7-605 may be placed nearly anywhere within or attached to the bit stream related to incoming media 7-499.

Indicator 7-605 may be comprised of various indicators, e.g., a level 0 indicator, a level 1 indicator, and a level 2 indicator, in one embodiment of the present invention. In the present embodiment, a level 0 indicator may be for indicating to CCM 7-300 that copying is permitted without restriction, e.g., incoming media 7-499 is not copyrighted or that the copyright is not asserted. In the present embodiment, a level 1 indicator may be for indicating to CCM 7-300 that one generation of copies of incoming media 7-499 may be made, such that incoming media 7-499 is an original copy and that one copy may be made. In the present embodiment, a level 2 indicator may be for indicating to CCM 7-300 that incoming media 7-499 is copyright protected and/or a copy thereof, and as such no digital copying is permitted.

For example, incoming media 7-499 is received by playback application 7-501. Application 7-501 detects an indicator 7-605 attached therewith, in this example, a level 2 bit is placed in the bit stream for indicating to CCM 7-300 that copying is not permitted.

For example, when CCM 7-300 is configured in system 7-210 such as that shown in Figure 7-5A, in response to a level 2 indicator bit, CCM 7-300, while controlling the audio path, then activates switches 7-311 and 7-511 to prevent any recording of incoming media 7-499.

When CCM 7-300 is configured in system 7-210 such as that shown in Figure 7-5B, in response to a level 2 indicator bit, CCM 7-300, while controlling the audio path, then activates switches 7-311 and 7-312 to prevent any recording of incoming media 7-499.

When CCM 7-300 is configured in system 7-210 such as that shown in Figure 7-5C, in response to a level 2 indicator bit, CCM 7-300, while controlling the audio path, then activates switches 7-311, 7-312, and 7-571 to prevent any recording of incoming media 7-499.

It is noted that CCM 7-300 can activate or deactivate switches coupled therewith, as described herein with reference to Figures 7-5A, 7-5B, 7-5C, and 7-5D, thereby funneling incoming media 7-499 through the secure media path, in this instance the audio path, to prevent unauthorized copying of incoming media 7-499. It is further noted that CCM 7-300 can detect media recording applications and devices as described herein, with reference to Figure 7-3.

Figures 7-7A, 7-7B, and 7-7C, are a flowchart 7-700 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 7-700 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 7-104 and/or computer usable non-volatile memory 7-103 of Figure 7-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 7-700, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 7-7A, 7-7B, and 7-7C. Within the present embodiment, it should be appreciated that the steps of flowchart 7-

700 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for restricting recording of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 7-300, can be installed in the client computer system 7-210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 7-700 is described in conjunction with Figures 7-2, 7-3, 7-4, 7-5A, 7-5B, 7-5C, and 7-5D, in order to more fully describe the operation of the present embodiment. In step 7-702 of Figure 7-7A, a user of a computer system, e.g., 7-210, causes the computer to communicatively couple to a web server, e.g., 7-250, via one or more communication networks, e.g., Internet 7-201, and proceeds to attempt to log in. It is

understood that the log in process of step 7-702 can be accomplished in a variety of ways in accordance with the present invention.

In step 7-704 of Figure 7-7A, web server 7-250 accesses a user database, e.g., 7-450, to determine whether the user and the computer system 7-210 logging in are registered with it. If the user and computer system 7-210 are registered with web server 7-250, the present embodiment proceeds to step 7-714. However, if the user and computer system 7-210 are logging in for the first time, web server 7-250 can initiate a user and computer system 7-210 registration process at step 7-706.

In step 7-706, registration of the user and computer system 7-210 is initiated. The user and computer system registration process can involve the user of computer system 7-210 providing personal information including, but not limited to, their name, address, phone number, credit card number, and the like. Web server 7-250 can verify the accuracy of the information provided. Web server 7-250 can also acquire information regarding the user's computer system 7-210 including, but not limited to, identification of media players disposed and operable on system 7-210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address. Additionally, web server 7-250 can further request that the user of computer system 7-210 to select a username and password.

In step 7-708 of Figure 7-7A, subsequent to the completion of the registration process, web server 7-250 generates a unique user identification (ID) or user key associated with the user of client computer system 7-210. The unique user ID, or user key, is then stored by web server 7-250 in a manner that is associated with that registered user. Furthermore, one or

more cookies containing that information specific to that user and the user's computer system 7-210, is installed in a non-volatile memory device, e.g., 7-103 and/or data storage device 7-108 of computer system 7-210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 7-210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user key, has been previously generated for the user and computer 7-210 that initially logged-in at step 7-702, the present embodiment proceeds to step 7-714

In step 7-710, web server 7-250 verifies that the user ID and the cookie(s) are properly installed in computer system 7-210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 7-250 can re-initiate the registration process at step 7-706. Alternatively, web server 7-250 can decouple computer system 7-210 from the network, thereby requiring a re-log in by the user of computer 7-210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 7-712.

In step 7-712 of Figure 7-7A, web server 7-250 can install a version of a copyright compliance mechanism, e.g., 7-300, onto one or more non-volatile memory devices of computer system 7-210. Installing CCM 7-300 into user's computer system 7-210 can facilitate client side compliance with licensing agreements and copyright restrictions applicable to specific delivered copyrighted media content. At step 7-712, the components of CCM 7-300, such as instructions 7-301, coder/decoder (codec) 7-303, agent programs 7-304, system hooks 7-305, skins 7-306, and custom media device drivers 7-307 (e.g., custom media device 7-310 of Figures 7-5B, 7-5C, and 7-5D), are installed in computer system 7-210, such

as that shown in Figures 7-5A, 7-5B, 7-5C, and 7-5D. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 7-300 into computer system 7-210. However, step 7-712 is well suited to install CCM 7-300 on computer system 7-210 in a wide variety of ways in accordance with the present embodiment. For example, CCM 7-300 can be installed as an integrated component within a media player application, media recorder application, and/or media player/recorder applications. Alternatively, CCM 7-300 can be installed as a stand alone mechanism within a client computer system 7-210. Additionally, CCM 7-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD, and/or as part of an installation package. In another embodiment, CCM 7-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD, reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 7-300 may be installed on client system 7-210 in a clandestine manner, relative to a user.

In step 7-714, web server 7-250 can request the previously established username and password of the user of client computer system 7-210. Accordingly, the user of client computer system 7-210 causes it to transmit to web server 7-250 the previously established username and password. Upon the receipt thereof, web server 7-250 may access a user database, e.g., 7-450, to determine their validity. If the username and password are invalid, web server 7-250 refuses access wherein flowchart 7-500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 7-716.

In step 7-716 of Figure 7-7A, web server 7-250 can access media file database 7-450 to determine if copyright compliance mechanism 7-300 has been updated to reflect changes

made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 7-300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 7-300 can provide compliance with current copyright restrictions.

Continuing with step 7-716, if web server 7-250 determines that CCM 7-300, or components thereof, of computer 7-210 has been updated, web server 7-250 initiates installation of the newer components and/or the most current version of CCM 7-300 into computer system 7-210, shown as step 7-718. If web server 7-250 determines that the current version of CCM 7-300 installed on system 7-210 does not have to be updated, the present embodiment proceeds to step 7-720 of Figure 7-7B.

In step 7-720 of Figure 7-7B, the user of client computer system 7-210 causes it to transmit to web server 7-250, e.g., via Internet 7-201, a request for a play list of available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g., 7-251.

In step 7-722, in response to web server 7-250 receiving the play list request, web server 7-250 transmits to client computer system 7-210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media content play list in a manner invisible to the user. It is noted that the

media content in content server 7-251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, and the like. The media content play list of step 7-720 can be implemented in diverse ways. In one example, web server 7-250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media content titles, or different lists of titles, can be loaded from content server 7-251 and passed to a CGI (common gateway interface) program operating on web server 7-250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 7-210. It is understood that the CGI can be written in nearly any software computing language.

In step 7-724 of Figure 7-7B, the user of client computer system 7-210 can utilize the received media content play list in conjunction with a media player application in order to cause client computer system 7-210 to transmit a request to web server 7-250 for delivery of desired media content, and wherein the user ID is automatically included therewith. The media content play list provided to client computer system 7-210 by web server 7-250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to client computer system 7-250 and the order in which the content will be delivered. Additionally, the user of client computer system 7-250 can create one or more customized play lists and store those play lists in system 7-250 and/or within web server 7-250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions

thereof. In another embodiment, the received media content play list can include a random media content delivery choice that the user of client computer system 7-210 can transmit to web server 7-250, with the user ID, to request delivery of the media content in a random manner.

In step 7-726, upon receiving the request for media content from client computer system 7-210, web server 7-250 determines whether the requesting media application operating on client computer system 7-210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 7-250 determines that the media application operating on system 7-210 is not a valid media application, the present embodiment proceeds to step 7-727 which in one embodiment, redirects client computer system 7-210 to a web site where the user of system 7-210 can download a valid media player application or to a software application which can identify client computer system 7-210, log system 7-210 out of web server 7-250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 7-250 determines that the media application operating on system 7-210 is a valid media application, the present embodiment proceeds to step 7-728.

In step 7-728 of Figure 7-7B, the present embodiment causes web server 7-250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer system 7-210 is valid. If web server 7-250 determines that the user ID is invalid, the present embodiment proceeds to step 7-729 where client computer system 7-210 can be logged off web server 7-250 or client computer system 7-210 can be returned to step

7-706 (of Figure 7-7A) to re-register and to have another unique user ID generated by web server 7-250. It is noted that the order in which steps 7-726 and 7-728 are performed can be altered such that step 7-728 can be performed prior to step 7-726. If web server 7-250 determines that the user ID is valid, the present embodiment proceeds to step 7-730.

In step 7-730, prior to web server 7-250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 7-732, CCM 7-300 governs certain media player applications and/or functions thereof that are operable on client computer system 7-210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 7-300 can utilize system hooks 7-305 to accomplish the functionality of step 7-730.

In step 7-732 of Figure 7-7C, the present embodiment causes web server 7-250 to transmit to client computer system 7-210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 7-210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 7-251. The address is time sensitive because, in one embodiment, the content server 7-251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 7-251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is

stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 7-251 that are actively coupled to one or more client computer systems (e.g., 7-210 to 7-230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client computer systems are uncoupled from the media content source of content server 7-251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 7-210 to content server 7-251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 7-251), or by routing a requesting client computer system (e.g., 7-210, 7-220, or 7-230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 7-251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 7-210 to 7-230.

The functionality of step 7-732 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 7-210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

In step 7-734 of Figure 7-7C, upon receiving the redirection command, the present embodiment causes the media playback application 7-501 (Figures 7-5A, 7-5B, 7-5C, and 7-

5D) operating on client computer system 7-210 to automatically transmit to content server 7-251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

In step 7-726 of Figure 7-7C, content server 7-251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 7-251 determines that the time sensitive access key is valid, the present embodiment proceeds to step 7-738 of Figure 7-7C. However, if content server 7-251 determines that the time access key is not valid, the present embodiment proceeds to step 7-737, a client redirect.

In step 7-737, content server redirects client computer 7-210 to step 7-732 (not shown) where a new access key is generated. Alternatively, step 7-737 causes the present embodiment to return to step 7-704 of Figure 7-7A. In yet another embodiment, step 7-737 causes client computer system 7-210 to be disconnected from content server 7-251.

In step 7-738 of Figure 7-7C, content server 7-251 transmits the requested high fidelity media content to client computer system 7-210. It is noted that each media content file delivered to client computer system 7-210 can have a header attached thereto, prior to delivery, as described with reference to Figure 7-4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can be utilized.

Still referring to step 7-738, content server 7-251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 7-210 as fast as the network transfer rate allows. Further, content server 7-251 can have its download rate adapted to be equal to the transfer rate of the network to which it is coupled. In another embodiment, the content server 7-251 download rate can be adapted to equal the network transfer rate of the client computer system 7-210 to which the media content is being delivered. For example, if client computer system 7-210 is coupled to Internet 7-201 via a T1 connection, then content server 7-251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 7-210, content server 7-251 is then able to transmit requested media content to another client computer system, e.g., 7-220 or 7-230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 7-250 to client computer system 7-210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 7-300 to client 7-210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 7-251 can transmit to client computer system 7-250 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 7-740 of Figure 7-7C, upon receiving the requested high fidelity media content from content server 7-251, the present embodiment causes client computer system 7-210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content is stored in client computer system 7-210 in a manner that restricts unauthorized redistribution. For example, the present embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time. Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory device, e.g., 7-103 or data storage device 7-108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is further noted that in one embodiment, when a user of client computer system 7-210 turns the computer off or causes client computer system 7-210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

Still referring to step 7-740, in another embodiment, the present embodiment can cause client computer system 7-210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media content can be stored in a manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is noted that by storing media content with client computer

system 7-210 (when allowed by applicable licensing agreements and copyright restrictions), content server 7-251 does not need to redeliver the same media content to client computer system 7-210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

In step 7-742 of Figure 7-7C, the received media content file is then fed into a media player application (e.g., playback application 7-501 of Figures 7-5A, 7-5B, 7-5C, and 7-5D), which then runs it through a codec, e.g., coder/decoder 7-303 of CCM 7-300, in one embodiment. In response, coder/decoder 7-303 sends an authorization request to the server, e.g., 7-251, with attached authorization data, as described herein. In response to receiving codec's 7-303 authorization request, server 7-251 compares the received authorization data with that stored in server 7-251, and subsequently, the present embodiment proceeds to step 7-744.

In step 7-744, the server 7-251 responds with a pass or fail authorization. If server 7-251 responds with a fail, such that the received authorization data is invalid, the present method can proceed to step 7-745, where server 7-251 can, in one embodiment, notify the user of client system 7-210, e.g., by utilization of skin 7-306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented to the user of client computer system 7-210, thereby informing the user that the delivery failed. However, if the authorization data passes, the present method proceeds to step 7-746.

In step 7-746, server 7-251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file via media

playback application 7-501 of Figures 7-5A, 7-5B, 7-5C, and 7-5D. In one embodiment, a decryption key can be included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 7-748.

In step 7-748 of Figure 7-6C, subsequent to media file decryption, the media file may be passed through CCM 7-300, e.g., a coder/decoder 7-303, to a media player application operating on client computer system 7-210, e.g., playback application 7-501 of Figures 7-5A, 7-5B, 7-5C, and 7-5D, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be involved in order to experience the media content, e.g., skin 7-306 of Figure 7-3. Skin 7-306 may be implemented when CCM 7-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard media player can be utilized by client computer system 7-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, coder/decoder 7-303 will repeatedly ensure that CCM 7-300 rules are being enforced at any particular moment during media playback, shown as step 7-750.

In step 7-750, as the media file content is delivered to the media player application, e.g., media player application 7-501 of Figures 7-5A, 7-5B, 7-5C, and 7-5D, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 7-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 7-300. If the rules are not enforced, e.g., change due to a user opening up a recording application (e.g., Total Recorder or alternative application) the present method proceeds to step 7-751. If the rules, in accordance with CCM 7-300, are enforced, the present method then proceeds to step 7-752.

In step 7-751 of Figure 7-7C, if the rules according to CCM 7-300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In one embodiment, CCM 7-300 can selectively control switches 7-311 and 7-511 (Figure 7-5A) to prevent output of incoming media 7-499 (Figures 7-5A, 7-5B, 7-5C, and 7-5D) to a recording application 7-502 (Figures 7-5A, 7-5B, and 7-5C, via wave shim driver 7-309 and direct sound 7-504 respectively, thus preventing unauthorized recording of incoming media 7-499. In another embodiment, CCM 7-300 can selectively control switches 7-311 and 7-312 (Figure 7-5B) to prevent output of incoming media 7-499 to recording application 7-502 via wave shim driver 7-309 and custom media device 7-310, thus preventing unauthorized recording of incoming media 7-499. In yet another embodiment, CCM 7-300 can selectively control switches 7-311, 7-312, to not only prevent incoming media 7-499 from being recorded in an unauthorized manner but can also selectively control switch 7-571 (Figure 7-5C) to prevent unauthorized output of incoming media 7-499 via digital output 7-575 of media hardware output device 7-570. In yet another embodiment, CCM 7-300 can selectively control switches 7-311, 7-312, 7-571, and 7-511 to a prevent kernel streaming

mechanism 7-515, e.g., DirectKS of Figure 7-5D, which can establish a connection with media device driver 7-505 of Figure 7-5D, from capturing incoming media content and returning it to a recording application (e.g., 7-502) to create an unauthorized recording of the media content. In one embodiment, incoming media 7-499 may not be output from digital output 7-575. In another embodiment, incoming media 7-499 may be output via digital output 7-575 but in an inaudible manner, e.g., silence. In yet another embodiment, incoming media 7-499 be audible but recording functionality can be disabled, such that the media content cannot be recorded.

In step 7-752, if the rules are enforced in accordance with CCM 7-300, coder/decoder 7-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 7-210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the present method as step 7-748. While the newly retrieved portion is presented, embodiments of the present method then again perform step 7-750, then step 7-752 or 7-751, then step 7-748, then 7-750, etc., in a continual loop until the media file contents are presented in their entirety. Advantageously, by constantly monitoring playing media files, CCM 7-300 can detect undesired activities and enforce those rules defined by CCM 7-300.

Figure 7-8 is a diagram of an exemplary high-speed global media content delivery system 7-800, in accordance with one embodiment of the present invention. In one embodiment, system 7-800 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 7-210, 7-220, and/or 7-230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 7-800 includes a global delivery

network 7-802 that can include multiple content servers, e.g., 7-804, 7-806, 7-808, 7-810, 7-812, 7-814, and 7-816, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 7-804 to 7-816 can store a portion, a substantial portion, or the entire contents of a media content library that can be delivered to client computer systems via a network, e.g., Internet 7-201, or a WAN (wide area network). Accordingly, each of content server 7-804 to 7-816 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

For example, a media delivery point (MDP) 7-816, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 7-451, to client computer systems within the Asiatic regions of the world while a media delivery point 7-812, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 7-804 to 7-816 represents the location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 7-804 to 7-816 can located anywhere within the world, and as such are not limited to the cities shown in global network 7-802.

Still referring to Figure 7-8, it is further noted that global system 7-802 is described in conjunction with Figures 7-2, 7-3, 7-4, 7-5A to 7-5D, and 7-6, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 7-210 of Figure 7-2, interacting with a web

server, e.g., web server 7-250 of Figure 7-2, as described herein, web server 7-250, in one embodiment, can redirect client computer system 7-210 to receive the desired media content from an MDP (e.g., 7-804 to 7-816) based on one or more differing criteria.

For example, computer system 7-210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 7-250 which can be located anywhere in the world. It is noted that steps 7-702 to 7-730 of Figures 7-7A and 7-7B can then be performed as described herein such that the present embodiment proceeds to step 7-732 of Figure 7-7C. At step 7-732, the present embodiment can determine which media delivery points, e.g., 7-804, 7-806, 7-808, 7-810, 7-812, 7-814, or 7-816, can subsequently provide and deliver the desired media content to client computer system 7-210.

Still referring to Figure 7-8, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 7-210, e.g., media delivery point 7-816. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client computer system 7-210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 7-804 to 7-816) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 7-210 at step 7-732, web server 7-250 transmits to client computer system 7-210 a redirection command to media delivery point/content server 7-812 along with a time sensitive access key, also referred to as a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 7-210 to eventually receive the requested media content. Within system 7-800, the redirection command can include a time sensitive address of the media content location within media delivery point 7-812. Accordingly, the New York City media delivery point 7-812 can subsequently provide and deliver the desired media content to client computer system 7-210. It is noted that steps 7-732 to 7-742 and step 7-737 of Figure 7-7C can be performed by media delivery point 7-812 in a manner similar to content server 7-251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 7-804 to 7-816, to provide high fidelity media content to client computer systems, e.g., 7-210 to 7-230, located throughout the world, communication network systems of the Internet 7-201 do not become overly congested. Additionally, global network 7-802 can deliver media content to a larger number of client computer systems (e.g., 7-210 to 7-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 7-320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

Referring still to Figure 7-8, it is noted that media delivery points/content servers 7-804 to 7-816 of global network 7-802 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 7-804 to 7-816 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that

media delivery points 7-804 to 7-816 can be functionally coupled such that if one of them fails, another media delivery point can take over and fulfill its functionality. Additionally, one or more web servers similar to web server 7-250 can be coupled to global network 7-802 utilizing wired and/or wireless communication technologies.

Within system 7-800, content server/media delivery point 7-804 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 7-806 to 7-816 of global network 7-802 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 7-804.

Specifically, the web infrastructure of media delivery point 7-804 includes firewalls 7-818 and 7-820 which are each coupled to global network 7-802. Firewalls 7-818 and 7-820 can be coupled to global network 7-802 in diverse ways, e.g., utilizing wired and/or wireless communication technologies. Particularly, firewalls 7-818 and 7-820 can each be coupled to global network 7-702 via a 10/100 Ethernet handoff. However, system 7-800 is not limited in any fashion to this specific implementation. It is noted that firewalls 7-818 and 7-820 are implemented to prevent malicious users from accessing any part of the web infrastructure of media delivery point 7-804, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 7-840 coupled to device 7-836 while firewall 7-820 includes a device 7-838, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 7-842 coupled to device 7-838. Furthermore, DB server 7-840 is coupled with device 7-836 and DB server 7-842 is coupled with device 7-838.

Still referring to Figure 7-8, and within media delivery point 7-804, firewall 7-818 is coupled to a director device 7-822 which is coupled to internal web application server 7-826

and 7-828, and a hub server 7-830. Firewall 7-820 is coupled to a director 7-824 which is coupled to internal web application servers 7-826 and 7-828, and hub server 7-830. Hub server 7-830 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 7-780 is coupled to a data storage device 7-832 capable of storing media content. Data storage device 7-832 can be implemented in a variety of ways, e.g., as a RAID (redundant array of inexpensive/independent disks) appliance.

It is noted that media delivery points 7-804 to 7-816 can be implemented in any manner similar to content server 7-250 described herein. Additionally, media delivery points 7-804 to 7-816 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 7-100 of Figure 7-1.

Advantageously, by providing a copyright compliance mechanism, e.g., 7-300, which can be easily and readily installed in a client computer system, e.g., 7-210, embodiments of the present invention can be implemented to control access to, control the delivery of, and control the user's experience with media content subject to copyright restrictions and licensing agreements, fore example, as defined by the DMCA. Additionally, by closely associating a client computer system, e.g., 7-210, with the user thereof, and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

A method of preventing unauthorized recording of electronic media according to one embodiment is described. The method comprises activating a compliance mechanism in response to a client system receiving media content. The compliance mechanism is coupled to the client system. The client system has a media content presentation application operable

thereon and is coupled to the compliance mechanism. The method further comprises controlling a data path of a kernel-mode media device driver of the client system with the compliance mechanism upon detection of a kernel streaming mechanism operable on the client system. The present method further comprises directing the media content from the kernel-mode media device driver to a media device driver coupled with the compliance mechanism, via the data path, for selectively restricting output of the media content.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 8: METHOD AND SYSTEM FOR CONTROLLING ACCESS
OF MEDIA ON A MEDIA STORAGE DEVICE

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals

capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop,

handheld, and/or other portable electronic device. For purposes of the present application, the term "portable electronic device" is not intended to be limited solely to conventional handheld or portable computers. Instead, the term "portable electronic device" is also intended to include many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, satellite radios, portable video playback devices (digital projectors), personal video eyewear, and other personal digital devices. Additionally, embodiments of the present invention are also well suited for implementation with theater presentation systems for public and/or private presentation in theaters, auditoriums, convention centers, etc.

Figure 8-1 is a block diagram illustrating an exemplary computer system 8-100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 8-100 can be nearly any type of computing system or electronic computing device including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 8-100 and which are executed by a processor(s) of computer system 8-100, in one embodiment. When executed, the instructions cause computer system 8-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 8-100 of Figure 8-1 comprises an address/data bus 8-110 for communicating information, one or more central processors 8-101 coupled to bus

8-110 for processing information and instructions. Central processor(s) 8-101 can be a microprocessor or any alternative type of processor. Computer system 8-100 also includes a computer usable volatile memory 8-102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 8-110 for storing information and instructions for processor(s) 8-101. Computer system 8-100 further includes a computer usable non-volatile memory 8-103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc., coupled to bus 8-110 for storing static information and instructions for processor(s) 8-101. In one embodiment, non-volatile memory 8-103 can be removable.

System 8-100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 8-104 coupled to bus 8-110 for enabling computer 8-100 to interface with other electronic devices. Communication interface 8-104 can include wired and/or wireless communication functionality. For example, in one embodiment, communication interface 8-104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire (IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, a satellite link, an Internet feed, a cable modem, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device 8-104. In such an instance, communication interface 8-104 may include a DSL modem.

Computer 8-100 of Figure 8-1 can also include one or more computer usable data storage device(s) 8-108 coupled to bus 8-110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 8-108 can be a magnetic storage device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 8-108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks or random array of inexpensive discs) configuration. It is noted that data storage device 8-108 can be located internal and/or external of system 8-100 and communicatively coupled with system 8-100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 8-100. It is further noted that nearly any portable electronic device, e.g., device 8-100a, can also be communicatively coupled with system 8-100 via utilization of wired and/or wireless technology, thereby expanding the functionality of system 8-100.

System 8-100 can also include an optional display device 8-105 coupled to bus 8-110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 8-105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma display, a field emission display (FED), video eyewear, a projection device (e.g., an LCD or DLP projector, a movie theater

projection system, and the like.), or any other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 8-100 of Figure 8-1 further includes an optional alphanumeric input device 8-106 coupled to bus 8-110 for communicating information and command selections to processor(s) 8-101, in one embodiment. Alphanumeric input device 8-106 is coupled to bus 8-110 and includes alphanumeric and function keys. Also included in computer 8-100 is an optional cursor control device 8-107 coupled to bus 8-110 for communicating user input information and command selections to processor(s) 8-101. Cursor control device 8-107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, a optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 8-106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 8-100 is configured with such functionality.

Figure 8-2 is a block diagram of an exemplary network 8-200 in which embodiments of the present invention may be implemented. In one example, network 8-200 enables one or more authorized client computer systems (e.g., 8-210, 8-220, and 8-230), each of which are coupled to Internet 8-201, to receive media content from a media content server, e.g., 8-251, via the Internet 8-201 while preventing unauthorized client computer systems from accessing media stored in a database of content server 8-251.

Network 8-200 includes a web server 8-250 and a content server 8-251 which are communicatively coupled to Internet 8-201. Further, web server 8-250 and content server 8-251 can be communicatively coupled without utilizing Internet 8-201, as shown. Web server 8-250, content server 8-251, and client computers 8-210, 8-220, and 8-230 can communicate with each other. It is noted that computers and servers of network 8-200 are well suited to be communicatively coupled in various implementations. For example, web server 8-250, content server 8-251, and client computer systems 8-210, 8-220, and 8-230 of network 8-200 can be communicatively coupled via wired communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 8-2, it is noted that web server 8-250, content server 8-251, and client computer systems 8-210, 8-220 and 8-230 of network 8-200 can, in one embodiment, be each implemented in a manner similar to computer system 8-100 of Figure 8-1. However, the server and computer systems in network 8-200 are not limited to such implementation. Additionally, web server 8-250 and content server 8-251 can perform various functionalities within network 8-200. It is also noted that, in one embodiment, web server 8-250 and content server 8-251 can both be disposed on a single or a plurality of physical computer systems, e.g., computer system 8-100 of Figure 8-1.

Further, it is noted that network 8-200 can operate with and deliver any type of media content, (e.g., audio, video, multimedia, graphics, information, data, software

programs, etc.) in any format. In one example, content server 8-251 can provide audio and video files to client computers 8-210 to 8-230 via Internet 8-201.

Figure 8-3 is a block diagram of an exemplary copyright compliance mechanism (CCM) 8-300, for controlling distribution of, access to, and/or copyright compliance of media files, in accordance with an embodiment of the present invention. In one embodiment, CCM 8-300 contains one or more software components and instructions for enabling compliance with DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry association of America) licensing agreements regarding media files. Additionally, CCM 8-300's software components and instructions further enable compliance with international recording restrictions such as those defined by the IFPI (international federation of phonographic industry) and/or the ISRC (international standard recording industry) and/or other foreign or international recording associations and/or foreign or international licensing restrictions. In one embodiment, CCM 8-300 may be integrated into existing and/or newly developed media player and recorder applications. In another embodiment, CCM 8-300 may be implemented as stand alone but in conjunction with existing media player/recorder applications, such that CCM 8-300 is communicatively coupled to existing media player/recorder applications. Alternatively, CCM 8-300 can be installed as a stand alone mechanism within a client computer system 8-210. Additionally, CCM 8-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD (secure digital card), and/or as part of an installation package. In another example, CCM 8-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a

music CD, reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 8-300 may be installed on client system 8-210 in a clandestine manner, relative to a user.

There are currently two types of copyright licenses recognized by the digital millennium copyright act (DMCA) for the protection of broadcast copyrighted material. One of the broadcast copyright licenses is a compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one example, the period of time can be three hours.

The other type of broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright holder grants permission for a server, e.g., web server 8-250 and/or content server 8-251 of Figure 8-2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with "no save" license, meaning that the end user is unable to save or

store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user to save that particularly music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This is commonly referred to as an interactive with "save" license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

It is noted that the DMCA allows for the "perfect" reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording.

In one embodiment, CCM (copyright compliance mechanism) 8-300 can be stored in web server 8-250 and/or content server 8-251 of network 8-200 and is configured to be installed into each client computer system, e.g., 8-210, 8-220 and 8-230, enabled to access the media files stored within content server 8-251 and/or web server 8-250. Alternatively, copyright compliance mechanism 8-300 can be externally disposed and communicatively coupled with a client computer system 8-200 via, e.g., a portable media device 8-100a of Figure 8-1. In yet another embodiment, CCM 8-300 can be configured to be operable from a media storage device upon which media files may be disposed.

Copyright compliance mechanism 8-300 is configured to be operable while having portions of components, entire components, combinations of components,

disposed within one or more memory units and/or data storage devices of a computer system, e.g., 8-210, 8-220, and/or 8-230.

Additionally, portions of components, entire components and/or combinations of components of CCM 8-300 can be readily updated, e.g., via Internet 8-201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or licensing agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications, or to counteract subversive and/or hacker-like attempts to unlawfully obtain one or more media files.

Referring to Figure 8-3, in one embodiment, CCM 8-300 is shown to include instructions 8-301 for enabling client computer system 8-210 to interact with web server 8-250 and content server 8-251 of network 8-200. Instructions 8-301 enable client computer system 8-210 to interact with servers, e.g., 8-250 and 8-251 in a network, e.g., 8-200.

The copyright compliance mechanism 8-300 also includes, in one embodiment, a user ID generator 8-302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 8-210. In one embodiment, the user ID and the cookie(s) are installed in computer system 8-210 prior to installation of the remaining components of the copyright compliance mechanism 8-300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 8-250 before the remaining components of a CCM 8-300 can be installed, within one

embodiment of the present invention. Additionally, the user ID/user key can contain, but is not limited to, the user's name, the user's address, the user's credit card number, an online payment account number, a verified email address, and an identity (username) and password selected by the user.

Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 8-210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 8-210, e.g., a MAC (machine address code) address, an IP address, and/or the serial number of the central processing unit (CPU) operable on computer system 8-210 and other information specific to the user and the computer system operated by the user.

Additionally, in another embodiment, user biometrics may be combined with computer system 8-210 data and user data and incorporated into the generation of a user ID. Alternatively, biometric data may be used in a stand alone implementation in the generation of the user ID. Types of biometric data that may be utilized to provide a user ID and/or authorization may include, but is not limited to, fingerprint data, retinal scan data, handprint data, facial recognition data, and the like.

It is noted that the information regarding the client computer system, e.g., 8-210, the user of system 8-210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 8-210, web server 8-250 can determine when a user of one computer system, e.g., 8-210, has given their username and password to another user using another computer system, e.g., 8-220. Because the username, password, and the user's computer system 8-210 are closely associated, web server 8-250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 8-250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 8-210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 8-250. Web server 8-250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 8-3, copyright compliance mechanism 8-300 further includes one or more coder/decoders (codec) 8-303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 8-300. In the present embodiment, coder/decoder 8-303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 8-251.

In the present embodiment, coder/decoder 8-303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 8-300 rules are being enforced at any particular moment during

media playback. It is noted that differing coder/decoders 8-303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 8-3, copyright compliance mechanism 8-300 also includes one or more agent programs 8-304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 8-210, 8-220, or 8-230, a server, e.g., web server 8-250 and/or content server 8-251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 8-304 can also be configured to maintain system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 8-300 also includes one or more system hooks 8-305, in one embodiment of the present invention. A system hook 8-305 is, in one embodiment, a library that is installed in a computer system, e.g., 8-210, and intercepts system wide events. For example, a system hook 8-305, in conjunction

with skins 8-306, can govern certain properties and/or functionalities of media player applications operating within the client computer system, e.g., 8-210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during playing of a media file, the progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that codec 8-303 compares the information for the media player application operating in client computer system, e.g., 8-210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 8-300 described herein.

The following C++ source code is exemplary implementation of the portion of a codec 8-303 for performing media player application detection, in accordance with an embodiment of the present invention. In another embodiment, the following source code can be modified to detect kernel streaming mechanisms operable within client system 8-210.

```
int
IsRecorderPresent(TCHAR *    szAppClass,
                  TCHAR *    szProdName)
{
    TCHAR    szWndText[_MAX_PATH]; /* buffer to receive title string for
window */
    HWND     hWnd;                /* handle to target window for operation */
    int      nRetVal;             /* return value for operation */

    /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
        == 0)
    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
            != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
    else
    {
        /* attempt to locate window with specified class */
        if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
            != (HWND) 0)
        {
            /* attempt to retrieve title string for window */
            if ( GetWindowText(hWnd,
                              szWndText,
                              _MAX_PATH)
                != 0)
            {
                /* attempt to locate product name within title string */
                if ( _tcsstr(szWndText, szProdName)
```

```

        != (TCHAR *) 0)
    {
        /* indicate application found */
        nRetVal = 1;
    }
}

/* return to caller */
return nRetVal;
}

```

It is further noted that codec 8-303 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 8-210. For example, codec 8-303, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 8-210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data pathways that are commonly used for recording, such that the utilized data pathway is governed by the copyright compliance mechanism 8-300. In one embodiment, this can be performed within a driver shim, e.g., wave driver shim 8-309 of Figures 8-5A and 8-5B.

A driver shim can be utilized for nearly any software output device, e.g., a standard Windows™ waveform output device, e.g., Windows™ Media Player, or hardware output device, e.g., speakers or headphones. Client computer system 8-210 is configured such that the driver shim (e.g., 8-309 of Figures 8-5A, 8-5B, 8-5C, and 8-5D) will appear as the default waveform media device to client level application programs. Thus, requests for processing of waveform media input and/or output will pass through the driver shim prior to being forwarded to the actual

waveform audio driver, media device driver 8-505 of Figures 8-5A and 8-5B. Such waveform input/output suppression can be triggered by other components of CCM 8-300, e.g., agent 8-304, to be active when a recording operation is initiated by a client computer system, e.g., 8-210, during the play back of media files which are subject to the DMCA.

It is noted that alternative driver shims can be implemented for nearly any waveform output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices, video, graphic and/or alphanumeric media files and video input and output devices.

The following C++ source code is an exemplary implementation of a portion of a codec 8-303 and/or a custom media device driver 8-307 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```

DWORD
_stdcall
widMessage(UINT      uDevId,
            UINT      uMsg,
            DWORD     dwUser,
            DWORD     dwParam1,
            DWORD     dwParam2)
{
    BOOL      bSkip;      /* flag indicating operation to be skipped */
    HWND      hWndMon;    /* handle to main window for monitor */
    DWORD     dwRetVal;    /* return value for operation */

    /* initialize variables */
    bSkip = FALSE;

```

```

dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;

if (uMsg == WIDM_START)
{
    /* attempt to locate window for monitor application */
    if ( (hWndMon = FindMonitorWindow())
        != (HWND) 0)
    {
        /* obtain setting for driver */
        bDrvEnabled = ( SendMessage(hWndMon,
                                   uiRegMsg,
                                   0,
                                   0)
                      == 0)
                      ? FALSE : TRUE;
    }

    if (bDrvEnabled == TRUE)
    {
        /* indicate error in operation */
        dwRetVal = MMSYSERR_NOMEM;

        /* indicate operation to be skipped */
        bSkip = TRUE;
    }
}

if (bSkip == FALSE)
{
    /* invoke entry point for original driver */
    dwRetVal = CallWidMessage(uDevId, uMsg, dwUser, dwParam1,
dwParam2);
}

/* return to caller */
return dwRetVal;
}

```

It is noted that when properly configured, system hook 8-305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 8-210 to 8-230. In one embodiment, system hook 8-305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a

Microsoft Windows™ operating environment, and accordingly can be implemented in a variety of ways. However, it is also noted that system hook 8-305 can be readily adapted for implementation in alternative operating systems, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 8-3, copyright compliance mechanism 8-300 also includes one or more skins 8-306, which can be designed to be installed in a client computer system, e.g., 8-210 to 8-230. In one embodiment, skins 8-306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 8-306 are customizable interfaces that, in one embodiment, are displayed on a display device (e.g., 8-105) of computer system 8-210 and provide functionalities for user interaction of delivered media content. Additionally, skins 8-306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

Furthermore, when system hook 8-305 is unable to govern a function of the media player application operable on a client computer system, e.g., 8-210, such that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 8-306 can be implemented to provide compliance.

Differing skins 8-306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 8-306a may be configured for utilization with a media content file

protected under a non-interactive agreement (DMCA), such that skin 8-306a may not include a pause function, a stop function, a selector function, and/or a save function, etc.. Another skin, e.g., skin 8-306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with "no save" agreement (DMCA), such that skin 8-306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with "save" agreement, a save or a burn to CD function.

Still referring to Figure 8-3, it is further noted that in the present embodiment, each skin 8-306 can have a unique name and signature. In one embodiment, skin 8-306 can be implemented, in part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 8-306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD5 hash, without knowledge of the MD5 hash table, changing the name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance mechanism 8-300 verifies skin 8-306, MD5 hash tables advantageously provide a deterrent against modifications made to the skin.

In one embodiment, copyright compliance mechanism 8-300 also includes one or more custom media device driver(s) 8-307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 8-210, can be configured to utilize a custom media

device application, e.g., custom media device 8-310 (shown in Figure 8-5B, 8-5C, and 8-5D), to control unauthorized recording of media content files. A custom media device application can be, but is not limited to, a custom media audio device application for media files having sound content, a custom video device application for media files having graphical and/or alphanumeric content, etc. In one embodiment, custom media device 8-310 of Figure 8-5B is an emulation of the custom media device driver 8-307. With reference to audio media, the emulation is performed in a waveform audio driver associated with custom media device 8-310. Driver 8-307 is configured to receive a media file being outputted by system 8-210 prior to the media file being sent to a media output device, e.g., media output device 8-570, and/or a media output application, e.g., recording application 8-502. Examples of a media output device includes, but is not limited to, a video card for video files, a sound card for audio files, etc. Examples of a recording application can include, but is not limited to, CD burner applications for writing to another CDs, ripper applications which capture the media file and change the format of the media file, e.g., from a CD audio file to an .mpeg audio file, and/or a .wav file, and/or an ogg vorbis file, and various other media formats. In one embodiment, client computer system 8-210 is configured with a custom media device driver 8-307 emulating custom media device 8-310, and which is system 8-210's default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of skin 8-306 or a custom web based player application or as part of a CCM 8-300 installation bundle, for forcing or requiring system 8-210 to have driver 8-307 as the default driver.

Therefore, when a media content file is received by system 8-210 from server 8-251, the media content file is playable, provided the media content file passes through the custom media device application (e.g., 8-310 of Figure 8-5B), emulated by custom media device driver 8-307, prior to being outputted. However, if an alternative media player application is selected, delivered media files from server 8-251 will not play on system 8-210.

Thus, secured media player applications would issue a media request to the driver, e.g., 8-307, for the custom media device 8-310 which then performs necessary media input suppression, e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 8-307 to a Windows™ waveform audio driver operable on system 8-210, thus reducing instances of incompatibilities with existing media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 8-300, e.g., agents 8-304, system hooks 8-305, and skins 8-306, or a combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of

ways including, but limited to, those found at developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 8-210 being configured with a custom media device as the default device driver e.g., device 8-310 of Figures 8-5B, 8-5C, and 8-5D, emulated by a custom media device driver 8-307, those media player applications that require their particular device driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured music. Further advantageous is that an emulated custom media device provides no native support for those media player applications used as a recording mechanism, e.g., DirectSound capture, (direct sound 8-504 of Figures 8-5A, 8-5B, 8-5C, and 8-5D) etc., that are able to bypass user-mode drivers for most media devices. Additionally, by virtue of the media content being sent through device driver 8-307, thus effectively disabling unauthorized saving/recording of media files, in one embodiment, media files that are delivered in a secured delivery system do not have to be encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-encrypted media files utilizing less storage space and network resources than encrypted media files, networks having limited resources can utilize the functionalities of driver 8-307 of CCM 8-300 to provide compliance with copyright restrictions and/or licensing agreements applicable with a media content file without having the processing overhead of encrypted media files.

Figure 8-4 is an illustration of an exemplary system 8-400 for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention. Specifically, system 8-400 illustrates web server 8-250, content server 8-

251, or a combination of web server 8-250 and content server 8-251 installing a copyright compliance mechanism (e.g., 8-300) in a client's computer system (e.g., 8-210) for controlling media file distribution and controlling user access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 8-210 can communicatively couple with a network (e.g., 8-200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 8-250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 8-250 can initiate a registration process with the requesting client 8-210. Client registration can be accomplished in a variety of ways. For example, web server 8-250 may deliver to a client 8-210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 8-250 including, but not limited to, user's name, address, phone number, credit card number, online payment account number, biometric identification (e.g., fingerprint, retinal scan, etc.), verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 8-4, web server 8-250 can, in one embodiment, detect information related to the client's computer system, e.g., 8-210, and store that information in a user/media database 8-450. For example, web server 8-250 can detect a unique identifier of client computer system 8-210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network

interface card) of client computer system 8-210 or the MAC address of the network interface adapter integrated on the motherboard of system 8-210. It is understood that a NIC enables a client computer system 8-210 to access web server 8-250 via Internet 8-201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 8-250 can, in one embodiment, detect and store (also in database 8-450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 8-210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 8-450) within web server 8-250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 8-210, all or part of this information can be installed in client computer system 8-210. In one embodiment, client computer system 8-210 information can be in the form of a cookie. Web server 8-250 then verifies that the user and client computer system 8-210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 8-250 installs a copyright compliance mechanism (e.g., 8-300) into the client's computer system, e.g., 8-210, in one embodiment of the present invention. It is noted that web server 8-250 may not initiate installation of CCM 8-300 until the user ID, password, and client computer system 8-210 information is verified. A variety of common techniques can be employed to install an entire CCM 8-300, portions of components, entire components, and/or combinations or a

function of components. For example, copyright compliance mechanism 8-300 can be installed in a hidden directory within client computer system 8-210, thereby preventing unauthorized access to it. In one embodiment of the present invention, it is noted that unless CCM 8-300 is installed in client computer system 8-210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 8-250 and/or content server 8-251.

Referring still to Figure 8-4, upon completion of client registration and installation of CCM 8-300, client computer system 8-210 can then request a media play list or a plurality of play lists, etc. In response, web server 8-250 determines whether the user of client computer system 8-210 is authorized to receive the media play list associated with the request. In one embodiment, web server 8-250 can request the username and password. Alternatively, web server 8-250 can utilize user database 8-450 to verify that computer 8-210 is authorized to receive a media play list. If client computer 8-210 is not authorized, web server 8-250 can initiate client registration, as described herein. Additionally, web server 8-250 can disconnect computer 8-210 or redirect it to an alternative web site. Regardless, if the user and client computer system 8-210 are not authorized, web server 8-250 will not provide the requested play list to client computer system 8-210.

However, if client computer system 8-210 is authorized, web server 8-210 can check copyright compliance mechanism 8-300 within data base 8-450 to determine if it, or any of the components therein, have been updated since the last time client computer system 8-210 logged in to web server 8-250. If a component of CCM 8-300 has been updated, web server 8-250 can install the updated component and/or

a more current version of CCM 8-300 into client computer system 8-210, e.g., via Internet 8-201. If CCM 8-300 has not been updated, web server 8-250 can then deliver the requested media play list to system 8-210 via Internet 8-201 along with an appended user key or user identification (ID). It is noted that user database 8-450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 8-210. Subsequently, the user of client computer system 8-210 can utilize the received media play list in combination with the media player application operating on system 8-210 to transmit a delivery request for one or more desired pieces of media content from web server 8-250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 8-4, upon receiving the media content delivery request, web server 8-250 can then check the validity of the requesting media application and the attached user key. In one embodiment, web server 8-250 can utilize user database 8-450 to check their validity. If either or both are invalid, web server 8-250, in one embodiment, can redirect unauthorized client computer system 8-210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 8-300 verifies that skins 8-306 are installed in client computer system 8-210. Additionally, CCM 8-300 further verifies that system hook(s) 8-305 have been run or are running to govern certain functions of those media player applications operable within client computer system 8-210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 8-300 further diverts and/or redirects certain pathways that are commonly used for recording, e.g., driver 8-307 of Figure 8-5A, device 8-310 of Figure 8-5B, device 8-570 of Figure 8-5C, and driver 8-505 of Figure 8-5D. Once

CCM 8-300 has performed the above described functions, web server 8-250 then, in one embodiment, issues to the client computer 8-210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 8-210 receiving the redirect command from web server 8-250, the media player application operating on client computer system 8-210 automatically transmits a new request and the time sensitive access key to content server 8-251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 8-251. If invalid, unauthorized client computer 8-210 is redirected by content server 8-250 to protect against abuse of the system and unauthorized access to content server 8-251. If the time sensitive access key is valid, content server 8-251 retrieves the desired media content from content database 8-451 and delivers it to client computer system 8-210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 8-210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 8-210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 8-4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g.,

title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 8-300 that identifies the media file as originating from a content server 8-251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 8-300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or the entire CCM 8-300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to modification of copyright compliance mechanism 8-300, modified portions of, or the entire updated CCM 8-300 can easily be installed in client computer system 8-210 in a variety of ways. For example, the updated CCM 8-300 can be installed during client interaction with web server 8-250, during user log-in, and/or while client computer system 8-210 is receiving the keyed play list.

Referring still to Figure 8-4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 8-251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be stored in a database 8-450, content database 8-451 or in some combination of

databases 8-450 and 8-451. It is further noted that the messages being passed back and forth between client computer system 8-210 and web server 8-250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting with adds, public domain encryption programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be similarly encrypted. Furthermore, in another embodiment, the media files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 8-307.

Subsequent to media file decryption, the media file may be passed through CCM 8-300, e.g., a coder/decoder 8-303, to a media player application operating on client computer system 8-210, e.g. playback application 8-501 of Figures 8-5A, 8-5B, 8-5C, 8-5D, and 8-6A, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 8-306 of Figure 8-3. A skin 8-306 may be necessary when CCM 8-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client computer system 8-210 to experience the media content. Typically, many media

player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 8-303 will repeatedly ensure that CCM 8-300 rules are being enforced at any particular moment during media playback, shown as step 8-650 of Figure 8-6C.

As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 8-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 8-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These presentation stoppages are collectively shown as step 8-651 of Figure 8-6C.

If the rules, in accordance with CCM 8-300, are enforced, the codec/decoder 8-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 8-210. The newly retrieved portion of the media file is then

presented by the client's media player application. While the newly retrieved portion is presented, CCM 8-300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 8-300 can detect undesired activities and enforces those rules as defined by CCM 8-300.

Figure 8-5A is an exemplary logic/bit path block diagram 8-500A showing utilization of a wave shim driver, e.g., wave shim driver 8-309 of Figure 8-3, in conjunction with copyright compliance mechanism 8-300, for selectively controlling recording of copyrighted media received by a client computer system, e.g., system 8-210, in one embodiment of the present invention. Copyright compliance mechanism 8-300 is, in one embodiment, installed and operational on client system 8-210 in the manner described herein.

In one embodiment, a copyright compliance mechanism 8-300 is shown as being communicatively coupled with a media playback application 8-501 via connection 8-520. Therefore, CCM 8-300 is enabled to communicate with playback application 8-501. In one embodiment, CCM 8-300 can be integrated into a media playback application. CCM 8-300 is also coupled to and controls a selectable switch 8-311 in wave shim driver 8-309 (as described in Figure 8-3) via connection 8-522. CCM 8-300 is further coupled to and controls a selectable switch 8-511 in direct sound 8-504 via connection 8-521. Depending upon the copyright restrictions and

licensing agreements applicable to an incoming media file, e.g., 8-499, CCM 8-300 controls whether switches 8-311 and 8-511 are open (shown), thus preventing incoming media 8-499 from reaching a media recording application, or closed (not shown) to allow recording of incoming media 8-499.

For example, incoming media 8-499 may originate from a content server, e.g., 8-251, coupled to system 8-210. In another example, incoming media 8-499 may originate from a personal recording/electronic device, e.g., a MP3 player/recorder or similar device, coupled to system 8-210. Alternatively, incoming media 8-499 may originate from a magnetic, optical or alternative media storage device inserted into a media device player coupled to system 8-210, e.g., a CD or DVD inserted into a CD or DVD player, a hard disk in a hot swappable hard drive, an SD (secure digital card) inserted into a SD reader, and the like. In yet another example, incoming media 8-499 may originate from another media player application or media recording application. Incoming media 8-499 may also originate from, but is not limited to, a satellite radio feed (e.g., XM radio), a personal communication device (e.g., a mobile phone), a cable television radio input (e.g., DMX (digital music express)), a digital distribution and/or a public presentation source via a network, Internet or other communication connection, pay-per-view and/or pay-per-play system, or a set-top box. It is noted that incoming media 8-499 can originate from nearly any source that can be coupled to system 8-210. However, regardless of the source of incoming media 8-499, embodiments of the present invention, described herein, can prevent unauthorized recording of the media.

Figure 8-5A shows a media playback application 8-501, e.g., an audio, video, or other media player application, operable within system 8-210 and configured to receive incoming media 8-499. Playback application 8-501 can be a playback application provided by an operating system, e.g., Media Player for Windows™ by Microsoft, a freely distributed playback application downloadable from the Internet, e.g., RealPlayer or LiquidAudio, a playback application provided by a webcaster, e.g., PressPlay, or a playback application commercially available.

Figure 8-5A shows media device driver 8-505 which, in one implementation, may be a software driver for a sound card coupled to system 8-210 having a media output device 8-570, e.g., speakers or headphones, coupled therewith for media files having audio content. In another implementation, media device driver 8-505 may be a software driver for a video card coupled with a display device, e.g., 8-105, for displaying media files having alphanumeric and/or graphical content, and so on. With reference to audio files, it is well known that a majority of recording applications assume a computer system, e.g., 8-210, has a sound card disposed therein, providing full-duplex sound functionality to system 8-210. This means media output driver 8-505 can simultaneously cause playback and recording of incoming media files 8-499. For example, media device driver 8-505 can playback media 8-499 along wave-out line 8-539 to media output device 8-570 (e.g., speakers for audible playback) via wave-out line 8-580 while outputting media 8-499 on wave-out line 8-540 to eventually reach recording application 8-502.

For purposes of Figures 8-5A, 8-5B, 8-5C, and 8-5D, the terms wave-in line and wave-out line are referenced from the perspective of media device driver 8-505.

Additionally, for the most part, wave-in lines are downwardly depicted and wave-out lines are upwardly depicted in Figures 8-5A, 8-5B, 8-5C, and 8-5D.

Continuing with Figure 8-5A, playback application 8-501 is coupled with an operating system (O/S) multimedia subsystem 8-503 and direct sound 8-504 via wave-in lines 8-531 and 8-551 respectively. O/S multimedia subsystem 8-503 is coupled to a wave shim driver 8-309 via wave-in line 8-533 and wave-out line 8-546. O/S multimedia subsystem 8-503 is also coupled to a recording application 8-502 via wave-out line 8-548. Operating system (O/S) multimedia subsystem 8-503 can be any O/S multimedia subsystem, e.g., a Windows™ multimedia subsystem for system 8-210 operating under a Microsoft O/S, a QuickTime™ multimedia subsystem for system 8-210 operating under an Apple O/S, and so on. Playback application 8-501 is also coupled with direct sound 8-504 via wave-in line 8-551.

Direct sound 8-504, in one instance, may represent access to a hardware acceleration feature in a standard audio device, enabling lower level access to components within media device driver 8-505. In another instance, direct sound 8-504 may represent a path that can be used by a recording application, e.g., Total Recorder, that can be further configured to bypass the default device driver, e.g., media device driver 8-505 to capture incoming media 8-499 for recording. For example, direct sound 8-504 can be enabled to capture incoming media 8-499 via wave-in line 8-551 and unlawfully output media 8-499 to a recording application 8-502 via wave-out line 8-568, as well as media 8-499 eventually going to media device driver 8-505, the standard default driver.

Still referring to Figure 8-5A, wave shim driver 8-309 is coupled with media device driver 8-505 via wave-in line 8-537 and wave-out line 8-542. Media device driver 8-505 is coupled with direct sound 8-504 via wave-in line 8-553 which is shown to converge with wave-in line 8-537 at media device driver 8-505. Media device driver 8-505 is also coupled with direct sound 8-504 via wave-out line 8-566.

Wave-out lines 8-542 and 8-566 are shown to diverge from wave-out line 8-540 at media device driver 8-505 into separate paths. Wave-out line 8-542 feeds into wave shim driver 8-309 and wave-out line 8-566 feeds into direct sound 8-504. When selectable switch 8-311 and 8-511 are open (shown), incoming media 8-499 cannot flow to recording application 8-502, thus preventing unauthorized recording of it.

For example, incoming media 8-499 is received at playback application 8-501. Playback application 8-501 activates and communicates to CCM 8-300 regarding copyright restrictions and/or licensing agreements applicable to incoming media 8-499. If recording restrictions apply to media 8-499, CCM 8-300 can, in one embodiment, open switches 8-311 and 8-511, thereby blocking access to recording application 8-502, effectively preventing unauthorized recording of media 8-499. In one embodiment, CCM 8-300 can detect if system 8-210 is configured with direct sound 8-504 selected as the default driver to capture incoming media 8-499, via wave-in line 8-551, or a recording application is detected and/or a hardware accelerator is active, such that wave driver shim 8-309 can be bypassed by direct sound 8-504. Upon detection, CCM 8-300 can control switch 8-511 such that the output path, wave-out line 8-568, to recording application 8-502 is blocked. It is

further noted that CCM 8-300 can detect media recording applications and devices as described herein, with reference to Figure 8-3.

Alternatively, if media device driver 8-505 is selected as the default driver, incoming media 8-499 is output from playback application 8-501 to O/S multimedia subsystem 8-503 on wave-in line 8-531. From subsystem 8-503, media 8-499 is output to wave shim driver 8-309 via wave-in line 8-533. The wave shim driver 8-309 was described herein with reference to Figure 8-3. Media 8-499 is output from wave shim driver 8-309 to media device driver 8-505 via wave-in line 8-537. Once received by media device driver 8-505, media 8-499 can be output via wave-out line 8-539 to a media output device 8-570 coupled therewith via wave-out line 8-580. Additionally, media device driver 8-505 can simultaneously output media 8-499 on wave-out line 8-540 back to wave shim driver 8-309. Dependent upon recording restrictions applicable to media 8-499, CCM 8-300 can, in one embodiment, close switch 8-311 (not shown as closed), thereby allowing media 8-499 to be output from wave shim driver 8-309 to subsystem 8-503 (via wave-out line 8-546) and then to recording application 8-502 via wave-out line 8-548. Alternatively, CCM 8-300 can also open switch 8-311, thereby preventing media 8-499 from reaching recording application 8-502.

It is particularly noted that by virtue of CCM 8-300 controlling both switches 8-311 and 8-511, and therefore controlling wave-out line 8-548 and wave-out line 8-568 leading into recording application 8-502, incoming media files, e.g., media 8-499, can be prevented from being recorded in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements

related to the incoming media. It is also noted that embodiments of the present invention in no way interfere with or inhibit the playback of incoming media 8-499.

Figure 8-5B is an exemplary logic/bit path block diagram 8-500B of a client computer system, e.g., 8-210, configured with a copyright compliance mechanism 8-300 for preventing unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 8-300 is, in one embodiment, coupled with and operational on client system 8-210 in the manner with reference to Figures 8-4, 8-5A, 8-5C, 8-5D, 8-6, and 8-7.

Diagram 8-500B of Figure 8-5B is similar to diagram 8-500A of Figure 8-5A, with a few changes. Particularly, diagram 8-500B includes a custom media device 8-310 communicatively interposed between and coupled to O/S multimedia subsystem 8-503 and wave shim driver 8-309. Custom media device 8-310 is coupled to O/S multimedia subsystem via wave-in line 8-533 and wave-out line 8-546. Custom media device 8-310 is coupled with wave shim driver 8-309 via wave-in line 8-535 and wave-out line 8-544. Additionally, custom media device 8-310 is coupled with direct sound 8-504 via wave-in line 8-553 which converges with wave-in line 8-533 and wave-out line 8-566 which diverges from wave-out line 8-546, in one embodiment.

Also added to Figure 8-5B is a media hardware output device 8-570 that is coupled to media device hardware driver 8-505 via line 8-580. Media hardware output device 8-570 can be, but is not limited to, a sound card for audio playback, a video card for video, graphical, alphanumeric, etc, output, and the like.

In one embodiment, CCM 8-300 is communicatively coupled with playback application 8-501 via connection 8-520, waveform driver shim 8-309 via connection 8-522, and custom media device 8-310, via connection 8-521. CCM 8-300 is coupled to and controls a selectable switch 8-311 in waveform driver shim 8-309 via connection 8-522. CCM 8-300 is also coupled to and controls a selectable switch 8-312 in custom audio device 8-310 via connection 8-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 8-499, CCM 8-300 controls whether switches 8-311 and 8-312 are open (shown), thus preventing the incoming media 8-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 8-499.

Continuing with Figure 8-5B, direct sound 8-504 is shown coupled with custom media device 8-310 via wave-in line 8-553, instead of being coupled with media device driver 8-505 (Figure 8-5A). In one embodiment, custom audio device 8-310 mandates explicit selection through system 8-210, meaning that custom audio device 8-310 needs to be selected as a default driver of system 8-210. By virtue of having the selection of custom media device 8-310 as the default driver of system 8-210, the data path necessary for direct sound 8-504 to capture the media content is selectively closed.

For example, incoming media 8-499 originating from nearly any source with reference to Figure 8-5A is received by media playback application 8-501 of system 8-210. Playback application 8-501 communicates to CCM 8-300, via connection 8-

520, to determine whether incoming media 8-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 8-501 communicates with CCM 8-300 to control switch 8-311 and 8-312 accordingly. In the present example, recording of incoming media 8-499 would violate applicable restrictions and/or agreements and therefore switch 8-312 is in an open position, such that the output path to recording application 8-502, e.g., wave-out line 8-548 and/or wave-out line 8-568, is effectively blocked, thereby preventing unauthorized recording of media 8-499.

Alternatively, if media device driver 8-505 is selected as the default driver, incoming media 8-499 continues from O/S multimedia subsystem 8-503, through custom audio device 8-310, wave driver shim 8-309, and into media device driver 8-505 where media 8-499 can be simultaneously output to media output device 8-570 via line 8-580, and output on wave-out line 8-540 to wave-and outputted by media device driver 8-505 to wave shim driver 8-309 on wave-out line 8-542. However, by virtue of CCM 8-300 controlling switch 8-311, wave-out line 8-544 which eventually leads to recording application 8-502 is blocked, thus effectively preventing unauthorized recording of media 8-499.

It is particularly noted that by virtue of CCM 8-300 controlling both switches 8-311 and 8-312 and therefore controlling wave-out line 8-548 and wave-out line 8-568, any incoming media files, e.g., incoming media 8-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 8-5B, it is further noted that custom media device 8-310 allows for unfettered playback of incoming media 8-499. Additionally, at any time during playback of media 8-499, custom media device 8-310 can be dynamically activated by CCM 8-300.

Figure 8-5C is an exemplary logic/bit path block diagram 8-500C of a client computer system, e.g., 8-210, configured with a copyright compliance mechanism 8-300 for preventing unauthorized output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 8-300 is, in one embodiment, coupled with and operational on client system 8-210 in the manner with reference to Figures 8-4, 8-5A, 8-5B, 8-5D, 8-6, and 8-7.

Diagram 8-500C of Figure 8-5C is similar to diagram 8-500B of Figure 8-5B, with a few changes. Particularly, diagram 8-500C includes a media hardware output device 8-570 that is coupled with a media device driver 8-505. In one embodiment, media hardware output device 8-570 can be a S/PDIF (Sony/Phillips Digital Interface) card for providing multiple outputs, e.g., an analog output 8-573 and a digital output 8-575. An alternative media hardware output device providing similar digital output can also be implemented as device 8-570 including, but not limited to, a USB (universal serial bus) output device and/or an externally accessible USB port located on system 8-210, a FireWire (IEEE1394) output device and/or an externally accessible FireWire port located on system 8-210, with wireline or wireless functionality. In the present embodiment, media hardware output device 8-570 is shown to include a switch 8-571 controlled by CCM 8-300 via communication line 8-

523, similar to switches 8-311 and 8-312, for controlling output of incoming media 8-499.

In one embodiment, CCM 8-300 is communicatively coupled with playback application 8-501 via connection 8-520, waveform driver shim 8-309 via connection 8-522, custom media device 8-310, via connection 8-521, and media hardware output device 8-570 via connection 8-523. CCM 8-300 is coupled to and controls a selectable switch 8-311 in waveform driver shim 8-309 via connection 8-522. CCM 8-300 is also coupled to and controls a selectable switch 8-312 in custom audio device 8-310 via connection 8-521. CCM 8-300 is further coupled to and controls a selectable switch 8-571 in media hardware output device 8-570 via connection 8-523. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 8-499, CCM 8-300 controls whether switches 8-311 and 8-312 are open (shown), thus preventing the incoming media 8-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 8-499. Additionally, CCM 8-300 controls whether switch 8-571 is open (shown), thus preventing incoming media 8-499 from being output from digital output 8-575 of media hardware output device 8-570, or closed (not shown) to allow incoming media 8-499 to be output from media hardware output device 8-570.

By controlling media hardware output device 8-570, copyright compliance mechanism 8-300 can prevent unauthorized output of incoming media 8-499 to, e.g., a digital recording device that may be coupled with digital output 8-575 of media hardware output device 8-570. Accordingly, in one embodiment, CCM 8-300 is enabled to also detect digital recording devices that may be coupled to a digital

output line, e.g., 8-571, of a media hardware output device, e.g., 8-570. Examples of a digital recording device that can be coupled to media hardware output device 8-570 can include, but is not limited to, mini-disc recorders, MP3 recorders, personal digital recorders, digital recording devices coupled with multimedia systems, personal communication devices, set-top boxes, and/or nearly any digital device that can capture an incoming media 8-499 being output from a media hardware output device 8-570, e.g., a sound card.

Continuing with Figure 8-5C, direct sound 8-504 is shown coupled with custom media device 8-310 via wave-in line 8-553, instead of being coupled with media device driver 8-505 (Figure 8-5A). In one embodiment, custom audio device 8-310 mandates explicit selection through system 8-210, meaning that custom audio device 8-310 is needs to be selected as a default driver of system 8-210. By virtue of having the selection of custom media device 8-310 as the default driver of system 8-210, the data path necessary for direct sound 8-504 to capture the media content is selectively closed.

For example, incoming media 8-499 originating from nearly any source with reference to Figure 8-5A is received by media playback application 8-501 of system 8-210. Playback application 8-501 communicates to CCM 8-300, via connection 8-520, to determine whether incoming media 8-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 8-501 communicates with CCM 8-300 to control switch 8-311, 8-312, and 8-571 accordingly. In the present example, recording of incoming media 8-499 would violate applicable restrictions and/or agreements and therefore switch 8-312 is in an open position,

such that the output path to recording application 8-502, e.g., wave-out line 8-548 and/or wave-out line 8-568, is effectively blocked, thereby preventing unauthorized recording of media 8-499.

Alternatively, if media device driver 8-505 is selected as the default driver, incoming media 8-499 continues from O/S multimedia subsystem 8-503, through custom audio device 8-310, wave driver shim 8-309, and into media device driver 8-505 where media 8-499 can be simultaneously output to media output device 8-570 via line 8-580, and output on wave-out line 8-540 to wave-and outputted by media device driver 8-505 to wave shim driver 8-309 on wave-out line 8-542. However, by virtue of CCM 8-300 controlling switch 8-311, wave-out line 8-544 which eventually leads to recording application 8-502 is blocked, thus effectively preventing unauthorized recording of media 8-499.

It is particularly noted that by virtue of CCM 8-300 controlling both switches 8-311 and 8-312 and therefore controlling wave-out line 8-548 and wave-out line 8-568, any incoming media files, e.g., incoming media 8-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 8-5C, it is particularly noted that although CCM 8-300 can prevent unauthorized recording of incoming media 8-499 by controlling switches 8-311 and 8-312, thus preventing incoming media 8-499 from reaching recording application 8-502, controlling switches 8-311 and 8-312 do nothing to prevent incoming media 8-499 from being captured by a peripheral digital device, e.g., a

mini-disc recorder, etc., coupled to a digital output 8-575 of device 8-570. Thus, by also controlling the output, via digital output 8-575 of media hardware output device 8-570, through control of switch 8-571, CCM 8-300 can prevent unauthorized capturing of incoming media 8-499 during output, e.g., on a sound card for audio files, a video card for video and/or graphical files, regardless of whether incoming media 8-499 is received in a secure and encrypted manner. However, when switch 8-571 is in a closed position, incoming media 8-499 may be played back in an unfettered manner. Additionally, at any time during playback of media 8-499, switch 8-312 of custom media device 8-310, switch 8-311 of media device driver 8-309, and/or switch 8-571 of media hardware output device 8-570 can be dynamically activated by CCM 8-300.

Figure 8-5D is an exemplary logic/bit path block diagram 8-500D of a client computer system, e.g., 8-210, configured with a copyright compliance mechanism 8-300 for preventing unauthorized kernel based output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 8-300 is, in one embodiment, coupled with and operational on client system 8-210 in the manner described herein with reference to Figures 8-4, 8-5A, 8-5B, 8-5C, 8-6, and 8-7.

Diagram 8-500D of Figure 8-5D is similar to diagram 8-500C of Figure 8-5C, with some changes. Particularly, diagram 8-500D includes a kernel streaming mechanism 8-515, e.g., DirectKS, that is coupled with a media device driver 8-505. In one embodiment, DirectKS 8-515 can be used for establishing a direct connection with media device driver 8-505. In the present embodiment, media device driver 8-

505 is shown to include a switch 8-511 controlled by CCM 8-300 via communication line 8-524, that is similar to switches 8-311, 8-312, and 8-571, for controlling output of incoming media 8-499.

In one embodiment, CCM 8-300 is communicatively coupled with: playback application 8-501 via connection 8-520, waveform driver shim 8-309 via connection 8-522, custom media device 8-310, via connection 8-521, and media device driver 8-505 via connection 8-524. Specifically, CCM 8-300 is coupled to and controls a selectable switch 8-311 in waveform driver shim 8-309 via connection 8-522. CCM 8-300 is also coupled to and controls a selectable switch 8-312 in custom audio device 8-310 via connection 8-521. CCM 8-300 is further coupled to and controls a selectable switch 8-511 in media device driver 8-505 via connection 8-524.

Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 8-499, CCM 8-300 controls whether switches 8-311 and 8-312 are open (shown), thus preventing the incoming media 8-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 8-499. Additionally, CCM 8-300 controls whether switch 8-511 is open (shown), thus preventing incoming media 8-499 from being returned from media device driver 8-505 to playback application 8-501, where DirectKS 8-515 can capture incoming media 8-499 and redirect it to recording application 8-502 to create an unauthorized copy or recording of incoming media 8-499. CCM 8-300 can also control whether switch 8-511 is closed (not shown) to allow incoming media 8-499 to be returned to playback application 8-501, where DirectKS 8-515 can capture and redirect incoming media 8-499 to recording application 8-502.

DirectKS 8-515, in one embodiment, may represent a kernel streaming mechanism that is adapted to establish a direct connection with a media device driver 8-505 of an operating system operable on client computer system 8-210, enabling kernel level access to media device driver 8-505. A kernel streaming mechanism can be implemented for the purpose of precluding utilization of standard audio APIs (application programming interfaces) to play or record media content, with particular attention paid to those playback applications with low latency requirements. DirectKS 8-515 can bypass existing APIs and communicate with media device driver 8-505. DirectKS 8-515 can be readily adapted to work in conjunction with a playback application, e.g., 8-501, to capture and redirect incoming media 8-499 to recording application 8-502, via wave-out line 8-588. Accordingly, DirectKS 8-515 can be implemented to create unauthorized media recordings.

By controlling media device driver 8-505, copyright compliance mechanism 8-300 can prevent unauthorized output of incoming media 8-499 to, e.g., a digital recording device 8-529 that may be coupled with recording application 8-502. In one embodiment, media device driver 8-505 is configured through the kernel mixer (not shown) to control the data path. Additionally, in one embodiment, CCM 8-300 is enabled to also detect a kernel streaming mechanism 8-515 (e.g., DirectKS) that may be operable on client computer system 8-210, as described herein with reference to Figure 8-3.

In one embodiment, custom media device 8-310 mandates explicit selection through system 8-210, meaning that custom media device 8-310 is needs to be selected as a default driver of system 8-210. By virtue of having the selection of

custom media device 8-310 as the default driver of system 8-210, the data path necessary for direct sound 8-504 to capture the media content is selectively closed.

For example, incoming media 8-499 originating from nearly any source with reference to Figure 8-5A is received by media playback application 8-501 of system 8-210. Playback application 8-501 communicates to CCM 8-300, via connection 8-520, to determine whether incoming media 8-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 8-501 communicates with CCM 8-300 to control switches 8-311, 8-312, 8-571, and 8-511, accordingly. In the present example, recording of incoming media 8-499 would violate applicable restrictions and/or agreements and therefore switch 8-511 is in an open position, such that the output path to recording application 8-502, e.g., wave-out line 8-548 and/or wave-out line 8-568 and/or wave-out line 8-588, is effectively blocked, thereby preventing unauthorized recording of media 8-499.

Still referring to Figure 8-5D, it is particularly noted that although CCM 8-300 can prevent unauthorized recording of incoming media 8-499 by controlling switches 8-311, 8-312, and 8-571, thus preventing incoming media 8-499 from reaching recording application 8-502, controlling switches 8-311, 8-312, and 8-571, do nothing to prevent incoming media 8-499 from being returned to recording application 8-502 by a kernel streaming mechanism 8-515(e.g., DirectKS), which enables capturing and redirecting of incoming media 8-499 to recording application 8-502, via wave-out line 8-588. Thus, by also controlling switch 8-511 of media device driver 8-505, CCM 8-300 can prevent kernel streaming mechanism 8-515 from returning incoming media 8-499 to recording application 8-502, thereby preventing incoming media 8-

499 from being captured and redirected to recording application 8-502 in an attempt to create an unauthorized copy and/or recording of incoming media 8-499.

However, when switch 8-511 is in a closed position, incoming media 8-499 may be returned to a recording application 8-502, such that recording could be possible, provided recording does not violate copyright restrictions applicable to incoming media 8-499. Additionally, at any time during playback of media 8-499, switch 8-312 of custom media device 8-310, switch 8-311 of wave shim driver 8-309, and/or switch 8-511 of media device driver 8-505 can be dynamically activated by CCM 8-300.

Figure 8-6A is a block diagram of a media file, e.g., incoming media 8-499, adapted to be received by a playback application, e.g., 8-501 of Figures 8-5A, 8-5B, 8-5C, and 8-5D, configured with an indicator 8-605 for enabling incoming media 8-499 to comply with rules according to the SCMS (serial copy management system). When applicable to a media file, e.g., 8-499, the SCMS allows for one copy of a copyrighted media file to be made, but not for copies of copies to be made. Thus, if incoming media 8-499 can be captured by a recording application, e.g., 8-502 of Figures 8-5A, 8-5B, 8-5C, and/or 8-5D, and/or a recording device, e.g. 8-529, and/or a peripheral recording device and/or a recording application coupled to a digital output of a media hardware output device, e.g., digital output 8-575 of media hardware output device 8-570 of Figures 8-5B, 8-5C, and 8-5D, and/or a kernel streaming mechanism 8-515, e.g., DirectKS of Figure 8-5D, unauthorized copying and/or recording may be accomplished.

Playback application 8-501 is coupled with CCM 8-300 via communication line 8-520 in a manner analogous to Figures 8-5A, 8-5B, 8-5C, and/or 8-5D. Although not shown in Figure 8-6, it is noted that CCM 8-300 is also coupled to switches 8-311 and 8-511 as shown in Figure 8-5A, switches 8-311 and 8-312 in Figure 8-5B, switches 8-311, 8-312, and 8-571 in Figure 8-5C, and switches 8-312, 8-311, 8-571, and 8-511, in Figure 8-5D.

In one embodiment, an indicator 8-605 is attached to incoming media 8-499 for preventing unauthorized copying or recording in accordance with the SCMS. In one embodiment, indicator 8-605 can be a bit that may be transmitted prior to beginning the delivery of incoming media 8-499 to playback application 8-501. In another embodiment, indicator 8-605 may be placed at the beginning of the bit stream of incoming media 8-499. In another embodiment, indicator 8-605 may be placed within a frame period of incoming media 8-499, e.g., every fifth frame, or any other desired frame period. In another embodiment, indicator 8-605 may be transmitted at a particular time interval or intervals during delivery of the media file, e.g. incoming media 8-499. Thus, indicator 8-605 may be placed nearly anywhere within or attached to the bit stream related to incoming media 8-499.

Indicator 8-605 may be comprised of various indicators, e.g., a level 0 indicator, a level 1 indicator, and a level 2 indicator, in one embodiment of the present invention. In the present embodiment, a level 0 indicator may be for indicating to CCM 8-300 that copying is permitted without restriction, e.g., incoming media 8-499 is not copyrighted or that the copyright is not asserted. In the present embodiment, a level 1 indicator may be for indicating to CCM 8-300 that one

generation of copies of incoming media 8-499 may be made, such that incoming media 8-499 is an original copy and that one copy may be made. In the present embodiment, a level 2 indicator may be for indicating to CCM 8-300 that incoming media 8-499 is copyright protected and/or a copy thereof, and as such no digital copying is permitted.

For example, incoming media 8-499 is received by playback application 8-501. Application 8-501 detects an indicator 8-605 attached therewith, in this example, a level 2 bit is placed in the bit stream for indicating to CCM 8-300 that copying is not permitted.

For example, when CCM 8-300 is configured in system 8-210 such as that shown in Figure 8-5A, in response to a level 2 indicator bit, CCM 8-300, while controlling the audio path, then activates switches 8-311 and 8-511 to prevent any recording of incoming media 8-499.

When CCM 8-300 is configured in system 8-210 such as that shown in Figure 8-5B, in response to a level 2 indicator bit, CCM 8-300, while controlling the audio path, then activates switches 8-311 and 8-312 to prevent any recording of incoming media 8-499.

When CCM 8-300 is configured in system 8-210 such as that shown in Figure 8-5C, in response to a level 2 indicator bit, CCM 8-300, while controlling the audio path, then activates switches 8-311, 8-312, and 8-571 to prevent any recording of incoming media 8-499.

It is noted that CCM 8-300 can activate or deactivate switches coupled therewith, as described herein with reference to Figures 8-5A, 8-5B, 8-5C, and 8-5D, thereby funneling incoming media 8-499 through the secure media path, in this instance the audio path, to prevent unauthorized copying of incoming media 8-499. It is further noted that CCM 8-300 can detect media recording applications and devices as described herein, with reference to Figure 8-3.

Figures 8-7A, 8-7B, and 8-7C, are a flowchart 8-700 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 8-700 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 8-104 and/or computer usable non-volatile memory 8-103 of Figure 8-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 8-700, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 8-7A, 8-7B, and 8-7C. Within the present embodiment, it should be appreciated that the steps of flowchart 8-700 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for restricting recording of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 8-300, can be installed in the client computer system 8-210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 8-700 is described in conjunction with Figures 8-2, 8-3, 8-4, 8-5A, 8-5B, 8-5C, and 8-5D, in order to more fully describe the operation of the present embodiment. In step 8-702 of Figure 8-7A, a user of a computer system, e.g., 8-210, causes the computer to communicatively couple to a web server, e.g., 8-250, via one or more communication networks, e.g., Internet 8-201, and proceeds to

attempt to log in. It is understood that the log in process of step 8-702 can be accomplished in a variety of ways in accordance with the present invention.

In step 8-704 of Figure 8-7A, web server 8-250 accesses a user database, e.g., 8-450, to determine whether the user and the computer system 8-210 logging in are registered with it. If the user and computer system 8-210 are registered with web server 8-250, the present embodiment proceeds to step 8-714. However, if the user and computer system 8-210 are logging in for the first time, web server 8-250 can initiate a user and computer system 8-210 registration process at step 8-706.

In step 8-706, registration of the user and computer system 8-210 is initiated. The user and computer system registration process can involve the user of computer system 8-210 providing personal information including, but not limited to, their name, address, phone number, credit card number, online payment account number, biometric identification (e.g., fingerprint, retinal scan, etc.), and the like. Web server 8-250 can verify the accuracy of the information provided. Web server 8-250 can also acquire information regarding the user's computer system 8-210 including, but not limited to, identification of media players disposed and operable on system 8-210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address. Additionally, web server 8-250 can further request that the user of computer system 8-210 to select a username and password.

In step 8-708 of Figure 8-7A, subsequent to the completion of the registration process, web server 8-250 generates a unique user identification (ID) or user key

associated with the user of client computer system 8-210. The unique user ID, or user key, is then stored by web server 8-250 in a manner that is associated with that registered user. Furthermore, one or more cookies containing that information specific to that user and the user's computer system 8-210, is installed in a non-volatile memory device, e.g., 8-103 and/or data storage device 8-108 of computer system 8-210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 8-210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user key, has been previously generated for the user and computer 8-210 that initially logged-in at step 8-702, the present embodiment proceeds to step 8-714

In step 8-710, web server 8-250 verifies that the user ID and the cookie(s) are properly installed in computer system 8-210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 8-250 can re-initiate the registration process at step 8-706. Alternatively, web server 8-250 can decouple computer system 8-210 from the network, thereby requiring a re-log in by the user of computer 8-210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 8-712.

In step 8-712 of Figure 8-7A, web server 8-250 can install a version of a copyright compliance mechanism, e.g., 8-300, onto one or more non-volatile memory devices of computer system 8-210. Installing CCM 8-300 into user's computer system 8-210 can facilitate client side compliance with licensing agreements and

copyright restrictions applicable to specific delivered copyrighted media content. At step 8-712, the components of CCM 8-300, such as instructions 8-301, coder/decoder (codec) 8-303, agent programs 8-304, system hooks 8-305, skins 8-306, and custom media device drivers 8-307 (e.g., custom media device 8-310 of Figures 8-5B, 8-5C, and 8-5D), are installed in computer system 8-210, such as that shown in Figures 8-5A, 8-5B, 8-5C, and 8-5D. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 8-300 into computer system 8-210. However, step 8-712 is well suited to install CCM 8-300 on computer system 8-210 in a wide variety of ways in accordance with the present embodiment. For example, CCM 8-300 can be installed as an integrated component within a media player application, media recorder application, and/or media player/recorder applications. Alternatively, CCM 8-300 can be installed as a stand alone mechanism within a client computer system 8-210. Additionally, CCM 8-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD, and/or as part of an installation package. In another embodiment, CCM 8-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD, reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 8-300 may be installed on client system 8-210 in a clandestine manner, relative to a user.

In step 8-714, web server 8-250 can request the previously established username and password of the user of client computer system 8-210. Accordingly, the user of client computer system 8-210 causes it to transmit to web server 8-250 the previously established username and password. Upon the receipt thereof, web

server 8-250 may access a user database, e.g., 8-450, to determine their validity. If the username and password are invalid, web server 8-250 refuses access wherein flowchart 8-500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 8-716.

In step 8-716 of Figure 8-7A, web server 8-250 can access media file database 8-450 to determine if copyright compliance mechanism 8-300 has been updated to reflect changes made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 8-300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 8-300 can provide compliance with current copyright restrictions.

Continuing with step 8-716, if web server 8-250 determines that CCM 8-300, or components thereof, of computer 8-210 has been updated, web server 8-250 initiates installation of the newer components and/or the most current version of CCM 8-300 into computer system 8-210, shown as step 8-718. If web server 8-250 determines that the current version of CCM 8-300 installed on system 8-210 does not have to be updated, the present embodiment proceeds to step 8-720 of Figure 8-7B.

In step 8-720 of Figure 8-7B, the user of client computer system 8-210 causes it to transmit to web server 8-250, e.g., via Internet 8-201, a request for a play list of available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g., 8-251.

In step 8-722, in response to web server 8-250 receiving the play list request, web server 8-250 transmits to client computer system 8-210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media content play list in a manner invisible to the user. It is noted that the media content in content server 8-251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, software applications, and the like. The media content play list of step 8-720 can be implemented in diverse ways. In one example, web server 8-250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media content titles, or different lists of titles, can be loaded from content server 8-251 and passed to a CGI (common gateway interface) program operating on web server 8-250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 8-210. It is understood that the CGI can be written in nearly any software computing language.

In step 8-724 of Figure 8-7B, the user of client computer system 8-210 can utilize the received media content play list in conjunction with a media player application in order to cause client computer system 8-210 to transmit a request to web server 8-250 for delivery of desired media content, and wherein the user ID is

automatically included therewith. The media content play list provided to client computer system 8-210 by web server 8-250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to client computer system 8-250 and the order in which the content will be delivered. Additionally, the user of client computer system 8-250 can create one or more customized play lists and store those play lists in system 8-250 and/or within web server 8-250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions thereof. In another embodiment, the received media content play list can include a random media content delivery choice that the user of client computer system 8-210 can transmit to web server 8-250, with the user ID, to request delivery of the media content in a random manner.

In step 8-726, upon receiving the request for media content from client computer system 8-210, web server 8-250 determines whether the requesting media application operating on client computer system 8-210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 8-250 determines that the media application operating on system 8-210 is not a valid media application, the present embodiment proceeds to step 8-727 which in one embodiment, redirects client computer system 8-210 to a web site where the user of system 8-210 can download a valid media

player application or to a software application which can identify client computer system 8-210, log system 8-210 out of web server 8-250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 8-250 determines that the media application operating on system 8-210 is a valid media application, the present embodiment proceeds to step 8-728.

In step 8-728 of Figure 8-7B, the present embodiment causes web server 8-250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer system 8-210 is valid. If web server 8-250 determines that the user ID is invalid, the present embodiment proceeds to step 8-729 where client computer system 8-210 can be logged off web server 8-250 or client computer system 8-250 can be returned to step 8-706 (of Figure 8-7A) to re-register and to have another unique user ID generated by web server 8-250. It is noted that the order in which steps 8-726 and 8-728 are performed can be altered such that step 8-728 can be performed prior to step 8-726. If web server 8-250 determines that the user ID is valid, the present embodiment proceeds to step 8-730.

In step 8-730, prior to web server 8-250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 8-732, CCM 8-300 governs certain media player applications and/or functions thereof that are operable on client computer system 8-210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 8-300 can utilize system hooks 8-305 to accomplish the functionality of step 8-730.

In step 8-732 of Figure 8-7C, the present embodiment causes web server 8-250 to transmit to client computer system 8-210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 8-210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 8-251. The address is time sensitive because, in one embodiment, the content server 8-251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 8-251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 8-251 that are actively coupled to one or more client computer systems (e.g., 8-210 to 8-230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client computer systems are uncoupled from the media content source of content server 8-251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 8-210 to content server 8-251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 8-251), or by routing a requesting client computer system (e.g., 8-210, 8-220, or 8-230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 8-251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 8-210 to 8-230.

The functionality of step 8-732 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 8-210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

In step 8-734 of Figure 8-7C, upon receiving the redirection command, the present embodiment causes the media playback application 8-501 (Figures 8-5A, 8-5B, 8-5C, and 8-5D) operating on client computer system 8-210 to automatically transmit to content server 8-251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

In step 8-736 of Figure 8-7C, content server 8-251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 8-251 determines that the time sensitive access key is valid, the present embodiment proceeds to step 8-738 of Figure 8-7C. However, if content

server 8-251 determines that the time access key is not valid, the present embodiment proceeds to step 8-737, a client redirect.

In step 8-737, content server redirects client computer 8-210 to step 8-732 (not shown) where a new access key is generated. Alternatively, step 8-737 causes the present embodiment to return to step 8-704 of Figure 8-7A. In yet another embodiment, step 8-737 causes client computer system 8-210 to be disconnected from content server 8-251.

In step 8-738 of Figure 8-7C, content server 8-251 transmits the requested high fidelity media content to client computer system 8-210. It is noted that each media content file delivered to client computer system 8-210 can have a header attached thereto, prior to delivery, as described with reference to Figure 8-4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can be utilized.

Still referring to step 8-738, content server 8-251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 8-210 as fast as the network transfer rate allows. Further, content server 8-251 can have its download rate adapted to be equal to the transfer rate of the network to which it is coupled. In

another embodiment, the content server 8-251 download rate can be adapted to equal the network transfer rate of the client computer system 8-210 to which the media content is being delivered. For example, if client computer system 8-210 is coupled to Internet 8-201 via a T1 connection, then content server 8-251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 8-210, content server 8-251 is then able to transmit requested media content to another client computer system, e.g., 8-220 or 8-230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 8-251 to client computer system 8-210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 8-300 to client 8-210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 8-251 can transmit to client computer system 8-210 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 8-740 of Figure 8-7C, upon receiving the requested high fidelity media content from content server 8-251, the present embodiment causes client computer system 8-210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content is stored in client computer system 8-210 in a manner that restricts unauthorized redistribution. For example, the present

embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time.

Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory device, e.g., 8-103 or data storage device 8-108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is further noted that in one embodiment, when a user of client computer system 8-210 turns the computer off or causes client computer system 8-210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

Still referring to step 8-740, in another embodiment, the present embodiment can cause client computer system 8-210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media content can be stored in a manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is noted that by storing media content with client computer system 8-210 (when allowed by applicable licensing agreements and copyright restrictions), content server 8-251 does not need to redeliver the same media content to client computer

system 8-210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

In step 8-742 of Figure 8-7C, the received media content file is then fed into a media player application (e.g., playback application 8-501 of Figures 8-5A, 8-5B, 8-5C, and 8-5D), which then runs it through a codec, e.g., coder/decoder 8-303 of CCM 8-300, in one embodiment. In response, coder/decoder 8-303 sends an authorization request to the server, e.g., 8-251, with attached authorization data, as described herein. In response to receiving codec's 8-303 authorization request, server 8-251 compares the received authorization data with that stored in server 8-251, and subsequently, the present embodiment proceeds to step 8-744.

In step 8-744, the server 8-251 responds with a pass or fail authorization. If server 8-251 responds with a fail, such that the received authorization data is invalid, the present method can proceed to step 8-745, where server 8-251 can, in one embodiment, notify the user of client system 8-210, e.g., by utilization of skin 8-306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented to the user of client computer system 8-210, thereby informing the user that the delivery failed. However, if the authorization data passes, the present method proceeds to step 8-746.

In step 8-746, server 8-251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file via media playback application 8-501 of Figures 8-5A, 8-5B, 8-5C, and 8-

5D. In one embodiment, a decryption key can be included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 8-748.

In step 8-748 of Figure 8-7C, subsequent to media file decryption, the media file may be passed through CCM 8-300, e.g., a coder/decoder 8-303, to a media player application operating on client computer system 8-210, e.g., playback application 8-501 of Figures 8-5A, 8-5B, 8-5C, and 8-5D, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be involved in order to experience the media content, e.g., skin 8-306 of Figure 8-3. Skin 8-306 may be implemented when CCM 8-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard media player can be utilized by client computer system 8-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis,

coder/decoder 8-303 will repeatedly ensure that CCM 8-300 rules are being enforced at any particular moment during media playback, shown as step 8-750.

In step 8-750, as the media file content is delivered to the media player application, e.g., media player application 8-501 of Figures 8-5A, 8-5B, 8-5C, and 8-5D, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 8-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 8-300. If the rules are not enforced, e.g., change due to a user opening up a recording application (e.g., Total Recorder or alternative application) the present method proceeds to step 8-751. If the rules, in accordance with CCM 8-300, are enforced, the present method then proceeds to step 8-752.

In step 8-751 of Figure 8-7C, if the rules according to CCM 8-300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In one embodiment, CCM 8-300 can selectively control switches 8-311 and 8-511 (Figure 8-5A) to prevent output of incoming media 8-499 (Figures 8-5A, 8-5B, 8-5C, and 8-5D) to a recording application 8-502 (Figures 8-5A, 8-5B, and 8-5C, via wave shim driver 8-309 and direct sound 8-504 respectively, thus preventing unauthorized recording of incoming media 8-499. In another embodiment, CCM 8-300 can selectively control switches 8-311 and 8-312 (Figure 8-5B) to prevent output of incoming media 8-499 to recording application 8-502 via wave shim driver 8-309 and custom media device 8-310, thus preventing unauthorized recording of incoming media 8-499. In yet another embodiment, CCM 8-300 can selectively control switches 8-311, 8-312, to not only prevent incoming media 8-499 from being

recorded in an unauthorized manner but can also selectively control switch 8-571 (Figure 8-5C) to prevent unauthorized output of incoming media 8-499 via digital output 8-575 of media hardware output device 8-570. In yet another embodiment, CCM 8-300 can selectively control switches 8-311, 8-312, 8-571, and 8-511 to a prevent kernel streaming mechanism 8-515, e.g., DirectKS of Figure 8-5D, which can establish a connection with media device driver 8-505 of Figure 8-5D, from capturing incoming media content and returning it to a recording application (e.g., 8-502) to create an unauthorized recording of the media content. In one embodiment, incoming media 8-499 may not be output from digital output 8-575. In another embodiment, incoming media 8-499 may be output via digital output 8-575 but in an inaudible manner, e.g., silence. In yet another embodiment, incoming media 8-499 be audible but recording functionality can be disabled, such that the media content cannot be recorded.

In step 8-752, if the rules are enforced in accordance with CCM 8-300, coder/decoder 8-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 8-210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the present method as step 8-748. While the newly retrieved portion is presented, embodiments of the present method then again perform step 8-750, then step 8-752 or 8-751, then step 8-748, then 8-750, etc., in a continual loop until the media file contents are presented in their entirety. Advantageously, by constantly monitoring playing media files, CCM 8-300 can detect undesired activities and enforce those rules defined by CCM 8-300.

Figure 8-8 is a diagram of an exemplary high-speed global media content delivery system 8-800, in accordance with one embodiment of the present invention. In one embodiment, system 8-800 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 8-210, 8-220, and/or 8-230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 8-800 includes a global delivery network 8-802 that can include multiple content servers, e.g., 8-804, 8-806, 8-808, 8-810, 8-812, 8-814, and 8-816, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 8-804 to 8-816 can store a portion, a substantial portion, or the entire contents of a media content library that can be delivered to client computer systems via a network, e.g., Internet 8-201, or a WAN (wide area network). Accordingly, each of content server 8-804 to 8-816 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

For example, a media delivery point (MDP) 8-816, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 8-451, to client computer systems within the Asiatic regions of the world while a media delivery point 8-812, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 8-804 to 8-816 represents the

location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 8-804 to 8-816 can be located anywhere within the world, and as such are not limited to the cities shown in global network 8-802.

Still referring to Figure 8-8, it is further noted that global system 8-802 is described in conjunction with Figures 8-2, 8-3, 8-4, 8-5A to 8-5D, and 8-6, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 8-210 of Figure 8-2, interacting with a web server, e.g., web server 8-250 of Figure 8-2, as described herein, web server 8-250, in one embodiment, can redirect client computer system 8-210 to receive the desired media content from an MDP (e.g., 8-804 to 8-816) based on one or more differing criteria.

For example, computer system 8-210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 8-250 which can be located anywhere in the world. It is noted that steps 8-702 to 8-730 of Figures 8-7A and 8-7B can then be performed as described herein such that the present embodiment proceeds to step 8-732 of Figure 8-7C. At step 8-732, the present embodiment can determine which media delivery points, e.g., 8-804, 8-806, 8-808, 8-810, 8-812, 8-814, or 8-816, can subsequently provide and deliver the desired media content to client computer system 8-210.

Still referring to Figure 8-8, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media

content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 8-210, e.g., media delivery point 8-816. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client computer system 8-210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 8-804 to 8-816) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 8-210 at step 8-732, web server 8-250 transmits to client computer system 8-210 a redirection command to media delivery point/content server 8-812 along with a time sensitive access key, also referred to as a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 8-210 to eventually receive the requested media content. Within system 8-800, the redirection command can include a time sensitive address of the media content location within media delivery point 8-812. Accordingly, the New York City media delivery point 8-812 can subsequently provide and deliver the desired media content to client computer system 8-210. It is noted that steps 8-732 to 8-742 and step 8-737 of Figure 8-7C can be performed by media delivery point 8-812 in a manner similar to content server 8-251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 8-804 to 8-816, to provide high fidelity media content to client computer systems, e.g., 8-210 to 8-230, located throughout the world, communication network systems of the Internet 8-201 do not become overly congested. Additionally, global network 8-802 can deliver media content to a larger number of client computer systems (e.g., 8-210 to 8-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

Referring still to Figure 8-8, it is noted that media delivery points/content servers 8-804 to 8-816 of global network 8-802 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 8-804 to 8-816 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that media delivery points 8-804 to 8-816 can be functionally coupled such that if one of them fails, another media delivery point can take over and fulfill its functionality. Additionally, one or more web servers similar to web server 8-250 can be coupled to global network 8-802 utilizing wired and/or wireless communication technologies.

Within system 8-800, content server/media delivery point 8-804 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 8-806 to 8-816 of global network 8-802 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 8-804.

Specifically, the web infrastructure of media delivery point 8-804 includes firewalls 8-818 and 8-820 which are each coupled to global network 8-802. Firewalls 8-818 and 8-820 can be coupled to global network 8-802 in diverse ways, e.g., utilizing wired and/or wireless communication technologies. Particularly, firewalls 8-818 and 8-820 can each be coupled to global network 8-702 via a 10/100 Ethernet handoff. However, system 8-800 is not limited in any fashion to this specific implementation. It is noted that firewalls 8-818 and 8-820 are implemented to prevent malicious users from accessing any part of the web infrastructure of media del836, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 8-840 coupled to device 8-836 while firewall 8-820 includes a device 8-838, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 8-842 coupled to device 8-838. Furthermore, DB server 8-840 is coupled with device 8-838 and DB server 8-842 is coupled with device 8-836.

Still referring to Figure 8-8, and within media delivery point 8-804, firewall 8-818 is coupled to a director device 8-822 which is coupled to internal web application server 8-826 and 8-828, and a hub server 8-830. Firewall 8-820 is coupled to a director 8-824 which is coupled to internal web application servers 8-826 and 8-828, and hub server 8-830. Hub server 8-830 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 8-780 is coupled to a data storage device 8-832 capable of storing media content. Data storage device 8-832 can be implemented in a variety of ways, e.g., as a RAID (redundant array of inexpensive/independent disks) appliance.

It is noted that media delivery points 8-804 to 8-816 can be implemented in any manner similar to content server 8-250 described herein. Additionally, media delivery points 8-804 to 8-816 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 8-100 of Figure 8-1.

In another embodiment, CCM 8-300 can be adapted to be disposed on a media storage device, e.g., media storage device 8-999 of Figures 8-10 and 8-11. Media storage device 8-999 can be, but is not limited to, a CD, a DVD, or other optical or magnetic storage device. By virtue of disposing a version of CCM 8-300 on a media storage device 8-999, embodiments of the present invention can provide copy protection for audio, video, multimedia, graphics, information, data, software programs, and other forms of media that may contain copyrighted material and which may be disposed on a media storage device. Alternatively, CCM 8-300 can be adapted to be installed on a computer system, e.g., client computer system 8-210, via a media storage device 8-999 upon which it may be disposed.

Figure 8-9 is a block diagram of a copyright compliance mechanism/media storage device (CCM/MSD) 8-900, a version of CCM 8-300 adapted to be disposed on a media storage device, e.g., media storage device 8-999 of Figures 8-10 and 8-11. It is noted that CCM 8-300 in CCM/MSD 8-900 is analogous to CCM 8-300 as described in Figures 8-3, 8-4, 8-5A to 8-5D, 8-6A and 8-7A to 8-7C. Further, CCM/MSD 8-900 can be readily updated in accordance with global delivery system 8-800, as described in Figures 8-7A to 8-7C, and Figure 8-8.

In one embodiment, CCM/MSD 8-900 is adapted to provide stand-alone compliance with copyright restrictions and licensing agreements applicable to media files that may be disposed on a media storage device, e.g., media storage device 8-999. In another embodiment CCM/MSD 8-900 is adapted to be installed on a computer system, e.g., client computer system 8-210 to provide compliance with copyright restrictions and licensing agreements applicable to media files as described in Figures 8-3, 8-4, 8-5A to 8-5D, 8-6A and 8-7A to 8-7C.

Referring to Figure 8-9, CCM/MSD 8-900 includes an autorun protocol component 8-910 for invoking automatic installation of CCM 8-300. To deter users from attempts at defeating various features inherent to CCM 8-300, e.g., the autorun feature, CCM 8-300's monitoring program, agent program 8-304, verifies that those features that are to be operational are operational, and if not, CCM 8-300 prohibits the user from experiencing the contents of the media storage device.

If a user somehow defeats the autorun feature, and the user attempts to utilize an application to capture an image of the content, the application will make an image of the content on the media storage device, which also images the copyright protection contained thereon, and when the image is played, CCM 8-300 recognizes the copy protection is present, and CCM 8-300 will only allow the user to experience the content when authorized, once CCM 8-300 is installed.

By virtue of the protections as described above provided by CCM 8-300, users will be able to experience the content of the media storage device in the

content's original high quality format, thereby obviating the need to compress the media file used on client system 8-210.

Advantageously, the user will no longer need to suffer through poor quality output as a result of severely compressed media files.

It is noted that when adapted to be implemented in conjunction with a secure file format, meaning that the format of the file is, without proper authorization, non-morphogenic, embodiments of the present invention also provide effective compliance with copyright restrictions and licensing agreements with secure files formats. CCM 8-300 can control the types of file formats into which the media file can be transformed, e.g., .wav, .mp3, etc.

In one embodiment, the autorun feature associated with media storage device drive 8-1112 of client system 8-210 is activated and operational. Alternatively, a notice of required autorun activation within client system 8-210 may be displayed on the media storage device and/or the case in which the media storage device is stored.

In another embodiment, if CCM 8-300 is present or if the user is coupled to a server, then messages containing instructions on how to activate the autorun feature of client system 8-210 may be presented to the user.

In one embodiment autorun protocol component 8-910 can detect media storage device drives resident on a computer system, e.g., client computer system 8-210.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 8-910 for detecting media storage device drives residing and operable on client computer system 8-210, according to one embodiment of the present invention.

```

if ( (dwRetVal = GetLogicalDrives())
    != (DWORD) 0)
{
    /* initialize variables */
    dwMask = (DWORD) 1;

    /* initialize path to root of current drive */
    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <<= 1)
    {
        if ((dwRetVal & dwMask) != 0)
        {
            /* construct path to root of drive */
            szDrive[0] = (TCHAR) 'A' + nIndex;

            if (GetDriveType(szDrive) == DRIVE_CDROM)
            {
                MessageBox((HWND) 0,
                    _T("CD-ROM drive found."),
                    szDrive,
                    MB_OK);
            }
            else
            {
                /* clear bit at current position */
                dwRetVal &= (~dwMask);
            }
        }
    }
}

```

In another embodiment, autorun protocol component 8-910 can detect whether a media storage device containing media files has been inserted into a media storage device drive coupled with client computer system 8-210, e.g., drive 8-1112 of Figure 8-10. In another embodiment, CCM 8-300 can include instructions for monitoring media storage device drive 8-1112, and upon detection of drive activation, CCM 8-300 determines what type of media storage device has been inserted therein. Subsequently, CCM 8-300 can detect various triggers on the media storage device to invoke its protection, e.g., a hidden file on newer media storage devices and/or the copyright indicator bit on legacy media storage devices, obviating the need for autorun. Upon detection, CCM 8-300 can invoke the appropriate protection for the associated media file.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 8-910 for detecting a media storage device inserted in a media storage device drive residing and operable on client computer system 8-210, according to one embodiment of the present invention.

```

    /* set error mode for operation */
    uiErrMode = SetErrorMode(SEM_FAILCRITICALERRORS);

    /* initialize path to root of current drive */
    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <= 1)
    {
        if ((dwCDROMMask & dwMask) != 0)
        {
            /* construct path to root of drive */
            szDrive[0] = (TCHAR) 'A' + nIndex;

```

```

        if ( GetDiskFreeSpace(szDrive,
                                &dwSectors,
                                &dwBytes,
                                &dwClustersFree,
                                &dwClusters)
            != 0)
        {
            /* add bit for drive to mask */
            dwRetVal |= dwMask;
        }
    }

    /* restore original error mode */
    SetErrorMode(uiErrMode);

```

Additionally, autorun protocol component 8-910 can also detect changes in media, e.g., insertion of a different media storage device 8-999. Further, other media changes can be detected subsequent to adaptation of the source code including, but not limited to, detecting a previously accessed media file, detecting a previously inserted media storage device.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 8-910 for detecting a change in media, according to one embodiment of the present invention.

```

    /* initialize path to root of current drive */
    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <=<= 1)
    {
        /* check for presence of CD-ROM media in drive */
        if ((dwCurrMask & dwMask) != 0)
        {
            /* check if media previously in drive */
            if ((dwPrevMask & dwMask) == 0)

```

```

    {
        /* construct path to root of drive */
        szDrive[0] = (TCHAR) 'A' + nIndex;

        /* check for presence of marker on drive */
        if (IsMPBMarkerPresent(szDrive) != 0)
        {
            /* process autorun information present on drive */
            nRetVal = ProcessAutorun(szDrive);
        }
    }
}

```

Still referring to Figure 8-9, CCM/MSD 8-900 also includes a kernel level filter driver 8-920 for controlling a data input path of an operating system coupled with and operable on client computer system 8-210.

CCM/MSD 8-900 also includes a generalized filter driver 8-930 for controlling ripping and "burning" applications, e.g., Nero, Roxio, Exact Audio Copy, and others, thereby preventing such activities.

The following C++ source code is an exemplary implementation of a portion of generalized filter driver 8-930 for controlling ripping and burning applications that may be residing on and operable within client computer system 8-210, in accordance with one embodiment of the present invention.

```

bool    bDisabled;          /* flag indicating CD reads disabled */

/* initialize variables */
bDisabled = false;

if (bProtected == true)
{

```

```

        if (type == IRP_MJ_DEVICE_CONTROL)
        {
            ULONG ulIoControlCode = stack-
>Parameters.DeviceIoControl.IoControlCode;

            if (ulIoControlCode == IOCTL SCSI_PASS_THROUGH)
            {
                SCSI_PASS_THROUGH * pspt =
(SCSI_PASS_THROUGH *)
Irp->AssociatedIrp.SystemBuffer;

                if ( (pspt != NULL)
                    && (pspt->Cdb[0] == SCSIOP_READ_CD))
                {
                    pspt->DataTransferLength = 0;
                    pspt->ScsiStatus = 0;

                    bDisabled = true;
                }
            }
            else if (ulIoControlCode ==
IOCTL SCSI_PASS_THROUGH_DIRECT)
            {
                SCSI_PASS_THROUGH_DIRECT * psptd =
(SCSI_PASS_THROUGH_DIRECT *)
Irp->AssociatedIrp.SystemBuffer;

                if ( (psptd != NULL)
                    && (psptd->Cdb[0] == SCSIOP_READ_CD))
                {
                    psptd->DataTransferLength = 0;
                    psptd->ScsiStatus = 0;

                    bDisabled = true;
                }
            }
        }

        if (bDisabled == true)
        {
            /* complete current request */
            status = CompleteRequest(Irp, STATUS_SUCCESS, 0);
        }
        else
        {
            /* pass request down without additional processing */
            status = IoAcquireRemoveLock(&pdx->RemoveLock, Irp);
        }
    }
}

```

```
        if (!INT_SUCCESS(status))  
            return CompleteRequest(Irp, status, 0);  
  
        IoSkipCurrentIrpStackLocation(Irp);  
        status = IoCallDriver(pdx->LowerDeviceObject, Irp);  
        IoReleaseRemoveLock(&pdx->RemoveLock, Irp);  
    }
```

Still referring to Figure 8-9, CCM/MSD 8-900 includes a CCM 8-300, analogous to CCM 8-300 of Figure 8-3, that is adapted to be installed in client computer system 8-210 in the manner described herein.

In one embodiment, kernel level filter driver 8-920, generalized filter driver 8-930 and CCM 8-300 of CCM/MSD 8-900 are automatically installed on client computer system 8-210, subsequent to insertion of media storage device 8-999 into a media storage device drive, e.g., media storage device drive 8-1112 of Figures 8-10 and 8-11. Autorun protocol component 8-910, as described above, detects insertion of media storage device 8-999 into an appropriate drive, and initiates installation of the components, e.g., CCM 8-300, driver 8-920 and driver 8-930. In one embodiment, drivers 8-920 and 8-930 may be temporarily installed and may be deleted upon removal of media storage device 8-999 from media storage device drive 8-1112. In yet another embodiment, drivers 8-920 and 8-930 may be installed in hidden directories and/or files within client computer system 8-210. In another embodiment, some components of CCM 8-300 can remain installed on client system 8-210, e.g. the monitoring program (agent program 8-304). In still another embodiment, other components, e.g., the kernel level filter driver 8-920, can be

dynamically loaded and unloaded as necessary in accordance with copyright restrictions and licensing agreements applicable to the media file.

Embodiments of the present invention utilize software, e.g., CCM/MSD 8-900, that is placed on media storage device 8-999, in conjunction with controlling software CCM 8-300 installed on client computer system 8-210, and web server 8-250 and/or content server 8-251, wherein each component is communicatively coupled with the other via the Internet, thereby enabling dynamic updating of CCM 8-300 in the manner as described with reference to Figure 8-4, and steps 8-716 and 8-718 of Figures 8-7A to 8-7C.

In the present embodiment, CCM/MSD 8-900 provides a stand alone DRM that is far more sophisticated than existing DRM solutions. This is because CCM/MSD 8-900 goes into the data pathway of the operating system operable on client computer system 8-210 and obtains control of the data pathway, e.g., filter driver 8-1108 of Figure 8-11, rather than exploiting inefficiencies or errors in the computer system.

Figure 8-10 is a block diagram of a communicative environment 8-1000 for controlling unauthorized reproduction of protected media files disposed on a media storage device. Included in communicative environment 8-1000 is a media storage device drive 8-1112 coupled with a client computer system 8-210 via a data/address bus 8-110. Client computer system 8-210 is coupled with web server 8-250 and content server 8-251 via Internet 8-201. A media storage device 8-999, upon which a CCM/MSD 8-900 may be disposed, is inserted in media storage device drive 8-

1112. Autorun protocol component 8-910 detects the insertion and automatically invokes installation of CCM 8-300, kernel level filter driver 8-920 and generalized filter driver 8-930 from media storage device 8-999 into client computer system 8-210. Subsequent to installation, CCM 8-300 initiates a dynamic update with web server 8-250 and/or content server 8-251, via Internet 8-201. By installing CCM 8-300 on client computer system, agent program 8-304 (Figure 8-3) of CCM 8-300 is able to control the integrity of the software. Additionally, by conferring with servers 8-250 and/or 251 via Internet 8-201 online, the CCM 8-300 software version on media storage device 8-999 and installed on client computer system 8-210 can be updated when circumventions occur and kept current from platform to platform.

Advantageously, the monitoring mechanism of agent program 8-304 enables constant morphing of the version of CCM 8-300 disposed on media storage device 8-999 by communicating with server 8-250 and/or 8-260 and utilizing the dynamic update capabilities of global network 8-800 to readily update that which has been installed on client computer system 8-210, via media storage device 8-999.

In one embodiment, the installation is performed clandestine with respect to the user and is initiated by inserting media storage device 8-999 into an appropriate media storage device drive, e.g. a magnetic/optical disk drive or alternative device drive coupled with client system 8-210. If the user is not registered with CCM 8-300, as described herein with reference to Figure 8-4 and Figures 8-7A to 8-7C, once installed, CCM 8-300 initiates an update process with web server 8-250 and/or content server 8-251 to readily include updates that have been invoked subsequent to release of the media file on media storage device 8-999. By virtue of the dynamic

update capabilities of CCM 8-300, regardless of the version of CCM 8-300 on media storage device 8-999, CCM 8-300 provides compliance with copyright restrictions and licensing agreements applicable to the media file on media storage device 8-999. Advantageously, enabling dynamic adaptability of CCM 8-300 provides for continued interoperability with new and updated operating systems, advancements in electronic technology, communication technologies and protocols, and the like, ensuring the effectiveness of CCM 8-300 into the future.

In another embodiment, if the user is a registered user with global delivery system 8-800, CCM 8-300 can detect which version is most current. Accordingly, when the version existing on client system 8-210 is more current than the version (for install) on media storage device 8-999, CCM 8-300 can bypass the install process and present the contents contained on media storage device 8-999 to the user for them to experience.

Further advantageous, this technology is backward compatible with media storage device drives manufactured subsequent to 1982. Additionally, CCM 8-300 is compatible with media storage devices having a copyright indicator bit disposed thereon. The copyright indicator bit has been included on all CDs released since 1982.

In the present embodiment of Figure 8-10, the media file is not encrypted on media storage device 8-999. In one embodiment, if the media file is encrypted on computer 8-210, it can be decrypted on the computer 8-210. However, home players and/or stand alone media playing devices rarely include a decryption

mechanism, and to experience the music on a home machine, the music is conventionally not encrypted.

In one embodiment, an additional component of CCM 8-300 is that the trigger for agent program 8-304 may be the copyright bit indicator. This means when the copyright indicator bit is detected by CCM 8-300, the functions of CCM 8-300 are initiated. Alternatively, in another embodiment, when the copyright bit indicator is not detected, CCM 8-300 may remain in an un-invoked or idle state. If CCM 8-300 can detect the copyright bit indicator, CCM 8-300 can provide the appropriate compliance with regard to copyright restrictions and licensing agreements applicable to the media files.

In an alternative embodiment, a trigger control in the table of contents of a media storage device 8-999 includes instructions for triggering autorun protocol 8-910 of CCM/MSD 8-900 and can utilize the copyright indicator bit or alternative implementation to trigger the technology. In this manner, CCM 8-300 can control copyrighted works while public domain material can be experienced and reproduced at a user's discretion. Because autorun is problematic for media storage device manufacturers, embodiments of CCM/MSD 8-900 can include alternative autorun programs that perform analogous to autorun.

In another embodiment, CCM 8-300 can invoke its own proprietary player, e.g., custom media device 8-310 as described with reference to Figure 8-3, thus enabling increased control of copyright restrictions and/or licensing agreements applicable to the media. By invoking custom media device 8-310, CCM 8-300

enables user experience of the media while providing protection against unauthorized reproduction of the media disposed on media storage device 8-999.

In an alternative embodiment, the media files and the CCM/MSD 8-900 disposed on a media storage device 8-999 are encrypted. This implementation is particularly advantageous for demonstration (demo) versions of media files, beta test versions, and the like that may be disposed on media storage device 8-999. It is noted that the present embodiment is operable in an online environment, meaning that client computer system 8-210 is communicatively coupled with web server 8-250 and/or content server 8-251 to enable a user experience of the content on a demo version of media storage device 8-999. In this implementation, CCM 8-300 allows for specific plays for specific users, which can be controlled via a network, e.g., network 8-1000 of Figure 8-10, and server 8-250 and/or 8-251.

In another embodiment, CCM 8-300 can be implemented for demo and/or pre-release protection. In this embodiment, CCM 8-300 utilizes sophisticated encryption technology to encrypt the table of contents and CCM 8-300 with an associated decrypted key located on client computer system. Encrypting CCM 8-300 can also deter nefarious attempts to reverse engineer CCM 8-300. Decryption can be performed using an associated decryption key. Alternatively, decryption can be performed by a proprietary or custom media player application resident on demo media storage device, e.g., 8-999.

The content of media storage device 8-999 is encrypted, using various levels of encryption to provide protection levels commensurate with copyright holders

desires and required protection. For example, media storage device 8-999 is delivered to a user or critic for the purposes of review, the user inserts media storage device 8-999 into the appropriate storage device reader or connector coupled with the journalist's computer, and CCM 8-300 is installed on client system 8-200 in a manner clandestine to the user. Once installed, CCM 8-300 initiates a communication session with web server 8-250/content server 8-251, where content server 8-251 can provide authorization for the user to experience the media on media storage device 8-999.

Accordingly, if the user, to whom demo media storage device 8-999 had been released, had demo media storage device 8-999 stolen, or if the user allowed alternative parties try to experience the content of media storage device 8-999, the unauthorized party would have to try to crack the encryption keys and the encryption of the actual content of media storage device 8-999, consuming non-trivial amounts of time.

Thus, CCM 8-300 is able to control which users receive authorization to experience the media of media storage device 8-999, how many times the user may experience the media, and CCM 8-300 may also define a period of time until the media may no longer be accessible. This may enable copyright holders to release the content on an authorized media storage device, e.g., 8-999, prior to pirated copies flooding the market.

Accordingly, a demo media storage device 8-999 may be configured such that a first user may get a copy, a second user may get a copy, and if it is known that

the second user will share the demo with a third and a fourth user, then the known users would be enabled to experience the media. Advantageously, by virtue of defining which users can access and experience the media, any unauthorized sharing of the media by one of the authorized users can be readily detected, and further sharing or experiencing of the media may be halted. Additionally, because the authorized user shared the media in an unauthorized manner, in a worse case scenario, criminal charges could be filed against that user.

It is noted that placing CCM/MSD 8-900 on a media storage device, e.g., 8-999, so as to enable installation of CCM 8-300 on client system 8-210 is one manner in which CCM 8-300 can be installed on client system 8-210. An alternative manner in which CCM 8-300 can be installed on client computer system 8-210 is through "cross-pollination." For example, webcasters broadcast the media file to the user. The media file has a CCM 8-300 coupled with the media file, and upon downloading the media file onto client computer system 8-210, embodiments of the present invention enable the installation of CCM 8-300 onto client computer system 8-210. In another manner, CCM 8-300 is incorporated into and becomes part of an operating system operational on client system 8-210. Alternatively, laws are passed that mandate the inclusion of CCM 8-300 on each client computer system 8-210.

Figure 8-11 is an exemplary logic/bit path block diagram 8-1100 of a client computer system, e.g., 8-210, configured with a copyright compliance mechanism (CCM) 8-300 for preventing unauthorized reproduction of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 8-300 is, in one embodiment, coupled with and operational on client

system 8-210 in any manner described with reference to Figures 8-4, 8-5A to 8-5D, 8-6A, and 8-7A to 8-7C, 8-9, and 8-10.

Diagram 8-1100 of Figure 8-11 includes a media storage device media extraction/creation application 8-1102 communicatively coupled to operating system input/output subsystem 8-1104 via wave in line 8-1121 and wave out line 8-1138. Operating system input/output subsystem 8-1104 is coupled with media storage device class driver 8-1106 via wave in line 8-1123 and wave out line 8-1136. Media storage device class driver 8-1106 is coupled with filter driver 8-1108 via wave in line 8-1125 and wave out line 8-1134. Filter driver 8-1108 is coupled with media storage device port driver 8-1110 via wave in line 8-1127 and wave out line 8-1132. Filter driver 8-1108 is shown to include a switch 8-1111, controlled by CCM 8-300 via coupling 8-1160. Media storage device port driver 8-1110 is coupled with media storage device drive 8-1112 via wave line in 8-1129 and wave line out 8-1130. Media storage device 8-999, shown to include CCM/MSD 8-900 is receivable by media storage device drive 8-1112. Additionally, CCM 8-300 is coupled with operating system input/output subsystem 8-1104 via wave in line 8-1150 and wave out line 8-1151.

In one embodiment, CCM 8-300 is coupled to and controls selectable switch 8-1111 in filter driver 8-1108. Depending upon the copyright restrictions and/or licensing agreements applicable to a media file disposed on media storage device 8-999, CCM 8-300 controls whether switch 8-1111 is open (shown), thus preventing the media file from reaching media extraction/creation application 8-1102, or closed (not shown) so as to allow reproduction of the protected media file. Media

extraction/creation application 8-1102 can be a ripping or burning application such as Nero, Roxio, Exact Audio Copy, or other readily available application.

Continuing with Figure 8-11, media storage device 8-999 is received by media storage device drive 8-1112. CCM 8-300 determines whether media storage device 8-999 or media disposed thereon is protected by any copyright restrictions and/or licensing agreements, e.g., via detection of a copyright indicator bit. CCM 8-300 communicates with filter driver 8-1108 to control switch 8-1111 accordingly. In the present example, reproducing media storage device 8-999, and/or the contents thereon, would violate applicable restrictions and/or agreements and therefore switch 8-1111 is in an open position such that the output path to media extraction/creation application 8-1102, e.g., wave-out line 8-1138, is effectively blocked thereby preventing unauthorized reproduction of media storage device 8-999.

It is particularly noted that by virtue of CCM 8-300 controlling switch 8-1111, and therefore controlling wave-out line 8-1138, any incoming copyright protected media disposed on a media storage device 8-999 can be prevented from being reproduced in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Advantageously, as new secure or proprietary file formats are developed, CCM 8-300 can be readily adapted to be functional therewith. Further, CCM/MSD 8-900 can prevent users from making unauthorized reproductions of media files, recording, copying, ripping, burning, etc. By using kernel level filter drivers, e.g., filter driver 8-1108, and getting to a low enough level within the operating system

(OS) on client system 8-210, CCM 8-300 can detect particular applications and when they request media storage device drive 8-1112 to poll the media file for copying, ripping, etc., and disable the data input path. CCM 8-300, in this embodiment, deals with the input pathway.

In one embodiment, alternative applications that monitor the state of client computer system 8-210 can enable the autorun functionality of client computer system 8-210 or alternatively, invoke an automatic mechanism similar to autorun to ensure invocation of CCM 8-300 for compliance of copyright restrictions and/or licensing agreements applicable to media storage device 8-999 and/or the copyright protected media disposed thereon.

In one embodiment, CCM 8-300 can invoke a proprietary media player from media storage device 8-999, or activate a proprietary media player resident and operable on client computer system 8-210, or an alternative authorized media player resident on client computer system 8-210, as described herein with reference to Figure 8-3.

When media storage device 8-999 is a multisession device, e.g., a compact disk having a data session and a music session (audio tracks), and it is inserted into media storage device drive 8-1112, CCM 8-300 looks at the contents of the media storage device 8-999, and in some operating systems the audio tracks will not be displayed. Instead, the data session is shown, as is an autorun file, e.g., autorun protocol component 8-910, and upon clicking, invokes a player application. CCM 8-

300 can have a data session and files to which a user may not have access unless a player application is invoked.

In one embodiment, the player application could deposit a monitoring portion (e.g., agent program 8-304) on client system 8-210, which in one embodiment may reside on client computer system 8-210 subsequent to removal of media storage device 8-999 from media storage device drive 8-1112.

By virtue of content in a multisession media storage device 8-999, which may not be directly accessible to most player applications, at some point the player application will be invoked which can then install the CCM 8-300 into client system 8-210, according to one embodiment of the present invention.

In one embodiment, a proprietary media player application is stored on media storage device 8-999. However, it is not automatically invoked. Upon some user intervention, e.g., inserting media storage device 8-999 into media storage device drive 8-1112, the media player application is loaded onto client system 8-210 which has CCM 8-300 integrated therewith. Thus, CCM 8-300 is launched regardless of autorun being activated or not activated, and mandates the user to utilize the proprietary media player application to experience the content of the media files on the media storage device. 8-999.

In an alternative embodiment, client computer system 8-210 has autorun off, wherein it is common for the user to be unable to play a media file unless a proprietary media player application is invoked. Activating the proprietary media

player application can initiate an installation of those components of CCM 8-300 that are bypassed when autorun is not active.

Advantageously, by providing a copyright compliance mechanism, e.g., 8-300, which can be easily and readily installed on a client computer system, e.g., 8-210, embodiments of the present invention can be implemented to control access to, the delivery of, and the user's experience with media content subject to copyright restrictions and/or licensing agreements, for example, as defined by the DMCA. Additionally, by closely associating a client computer system, e.g., 8-210, with the user thereof and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

A method of preventing unauthorized reproduction of protected media disposed on a media storage device according to one embodiment is described. The method is comprised of activating an autorun mechanism disposed on said media storage device. The activating is in response to a device drive coupled with a computer system receiving the media storage device. The autorun mechanism for initiating installing a compliance mechanism on the computer system. The method further includes installing the compliance mechanism on the computer system. The compliance mechanism communicatively coupled with the computer system when installed thereon. The compliance mechanism is for enforcing compliance with a usage restriction applicable to the protected media. The method further includes obtaining control of a data input pathway operable on the computer system. The method further includes preventing the protected media on the media storage device

from being captured by an extractor mechanism via the data input pathway while enabling presentation of the protected media.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

Section 9: METHOD AND SYSTEM FOR CONTROLLING PRESENTATION OF
MEDIA ON A MEDIA STORAGE DEVICE

Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, to one of ordinary skill in the art, the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Some portions of the detailed description which follows are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital memory system. These descriptions and representations are the means used by those skilled in the data processing art to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those involving physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated

in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like, with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that discussions of the present invention refer to actions and processes of a computing system, or similar electronic computing device that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system's memories or registers, or other such information storage, transmission, or display devices.

In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. To one skilled in the art, the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the present invention.

Embodiments of the present invention are discussed primarily in the context of a network of computer systems such as a network of desktop, workstation, laptop, handheld, and/or other portable electronic device. For purposes of the present

application, the term “portable electronic device” is not intended to be limited solely to conventional handheld or portable computers. Instead, the term “portable electronic device” is also intended to include many mobile electronic devices. Such mobile devices include, but are not limited to, portable CD players, MP3 players, mobile phones, portable recording devices, satellite radios, portable video playback devices (digital projectors), personal video eyewear, and other personal digital devices. Additionally, embodiments of the present invention are also well suited for implementation with theater presentation systems for public and/or private presentation in theaters, auditoriums, convention centers, etc.

Figure 9-1 is a block diagram illustrating an exemplary computer system 9-100 that can be used in accordance with an embodiment of the present invention. It is noted that computer system 9-100 can be nearly any type of computing system or electronic computing device including, but not limited to, a server computer, a desktop computer, a laptop computer, or other portable electronic device. Within the context of the present invention, certain discussed processes, procedures, and steps are realized as a series of instructions (e.g., a software program) that reside within computer system memory units of computer system 9-100 and which are executed by a processor(s) of computer system 9-100, in one embodiment. When executed, the instructions cause computer system 9-100 to perform specific actions and exhibit specific behavior which is described in detail herein.

Computer system 9-100 of Figure 9-1 comprises an address/data bus 9-110 for communicating information, one or more central processors 9-101 coupled to bus 9-110 for processing information and instructions. Central processor(s) 9-101 can

be a microprocessor or any alternative type of processor. Computer system 9-100 also includes a computer usable volatile memory 9-102, e.g., random access memory (RAM), static RAM (SRAM), dynamic RAM (DRAM), synchronous dynamic RAM (SDRAM), double data rate RAM (DDR RAM), etc., coupled to bus 9-110 for storing information and instructions for processor(s) 9-101. Computer system 9-100 further includes a computer usable non-volatile memory 9-103, e.g., read only memory (ROM), programmable ROM, electronically programmable ROM (EPROM), electrically erasable ROM (EEPROM), flash memory (a type of EEPROM), etc., coupled to bus 9-110 for storing static information and instructions for processor(s) 9-101. In one embodiment, non-volatile memory 9-103 can be removable.

System 9-100 also includes one or more signal generating and receiving devices, e.g., signal input/output device(s) 9-104 coupled to bus 9-110 for enabling computer 9-100 to interface with other electronic devices. Communication interface 9-104 can include wired and/or wireless communication functionality. For example, in one embodiment, communication interface 9-104 is a serial communication port, but can alternatively be one of a number of well known communication standards and protocols, e.g., a parallel port, an Ethernet adapter, a FireWire (IEEE 1394) interface, a Universal Serial Bus (USB), a small computer system interface (SCSI), an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, a satellite link, an Internet feed, a cable modem, and the like. In another embodiment, a digital subscriber line (DSL) can be implemented as signal input/output device 9-104. In such an instance, communication interface 9-104 may include a DSL modem.

Computer 9-100 of Figure 9-1 can also include one or more computer usable data storage device(s) 9-108 coupled to bus 9-110 for storing instructions and information, in one embodiment of the present invention. In one embodiment, data storage device 9-108 can be a magnetic storage device, e.g., a hard disk drive, a floppy disk drive, a zip drive, or other magnetic storage device. In another embodiment, data storage device 9-108 can be an optical storage device, e.g., a CD (compact disc), a DVD (digital versatile disc), or other alternative optical storage device. Alternatively, any combination of magnetic, optical, and alternative storage devices can be implemented, e.g., a RAID (random array of independent disks or random array of inexpensive discs) configuration. It is noted that data storage device 9-108 can be located internal and/or external of system 9-100 and communicatively coupled with system 9-100 utilizing wired and/or wireless communication technology, thereby providing expanded storage and functionality to system 9-100. It is further noted that nearly any portable electronic device, e.g., device 9-100a, can also be communicatively coupled with system 9-100 via utilization of wired and/or wireless technology, thereby expanding the functionality of system 9-100.

System 9-100 can also include an optional display device 9-105 coupled to bus 9-110 for displaying video, graphics, and/or alphanumeric characters. It is noted that display device 9-105 can be a CRT (cathode ray tube), a thin CRT (TCRT), a liquid crystal display (LCD), a plasma display, a field emission display (FED), video eyewear, a projection device (e.g., an LCD or DLP projector, a movie theater projection system, and the like.), or any other display device suitable for displaying video, graphics, and alphanumeric characters recognizable to a user.

Computer system 9-100 of Figure 9-1 further includes an optional alphanumeric input device 9-106 coupled to bus 9-110 for communicating information and command selections to processor(s) 9-101, in one embodiment. Alphanumeric input device 9-106 is coupled to bus 9-110 and includes alphanumeric and function keys. Also included in computer 9-100 is an optional cursor control device 9-107 coupled to bus 9-110 for communicating user input information and command selections to processor(s) 9-101. Cursor control device 9-107 can be implemented using a number of well known devices such as a mouse, a trackball, a track pad, a joy stick, a optical tracking device, a touch screen, etc. It is noted that a cursor can be directed and/or activated via input from alphanumeric input device 9-106 using special keys and key sequence commands. It is further noted that directing and/or activating the cursor can be accomplished by alternative means, e.g., voice activated commands, provided computer system 9-100 is configured with such functionality.

Figure 9-2 is a block diagram of an exemplary network 9-200 in which embodiments of the present invention may be implemented. In one example, network 9-200 enables one or more authorized client computer systems (e.g., 9-210, 9-220, and 9-230), each of which are coupled to Internet 9-201, to receive media content from a media content server, e.g., 9-251, via the Internet 9-201 while preventing unauthorized client computer systems from accessing media stored in a database of content server 9-251.

Network 9-200 includes a web server 9-250 and a content server 9-251 which are communicatively coupled to Internet 9-201. Further, web server 9-250 and content server 9-251 can be communicatively coupled without utilizing Internet 9-201, as shown. Web server 9-250, content server 9-251, and client computers 9-210, 9-220, and 9-230 can communicate with each other. It is noted that computers and servers of network 9-200 are well suited to be communicatively coupled in various implementations. For example, web server 9-250, content server 9-251, and client computer systems 9-210, 9-220, and 9-230 of network 9-200 can be communicatively coupled via wired communication technology, e.g., twisted pair cabling, fiber optics, coaxial cable, etc., or wireless communication technology, or a combination of wired and wireless communication technology.

Still referring to Figure 9-2, it is noted that web server 9-250, content server 9-251, and client computer systems 9-210, 9-220 and 9-230 of network 9-200 can, in one embodiment, be each implemented in a manner similar to computer system 9-100 of Figure 9-1. However, the server and computer systems in network 9-200 are not limited to such implementation. Additionally, web server 9-250 and content server 9-251 can perform various functionalities within network 9-200. It is also noted that, in one embodiment, web server 9-250 and content server 9-251 can both be disposed on a single or a plurality of physical computer systems, e.g., computer system 9-100 of Figure 9-1.

Further, it is noted that network 9-200 can operate with and deliver any type of media content, (e.g., audio, video, multimedia, graphics, information, data, software

programs, etc.) in any format. In one example, content server 9-251 can provide audio and video files to client computers 9-210 to 9-230 via Internet 9-201.

Figure 9-3 is a block diagram of an exemplary copyright compliance mechanism (CCM) 9-300, for controlling distribution of, access to, and/or copyright compliance of media files, in accordance with an embodiment of the present invention. In one embodiment, CCM 9-300 contains one or more software components and instructions for enabling compliance with DMCA (digital millennium copyright act) restrictions and/or RIAA (recording industry association of America) licensing agreements regarding media files. Additionally, CCM 9-300's software components and instructions further enable compliance with international recording restrictions such as those defined by the IFPI (international federation of phonographic industry) and/or the ISRC (international standard recording industry) and/or other foreign or international recording associations and/or foreign or international licensing restrictions. In one embodiment, CCM 9-300 may be integrated into existing and/or newly developed media player and recorder applications. In another embodiment, CCM 9-300 may be implemented as stand alone but in conjunction with existing media player/recorder applications, such that CCM 9-300 is communicatively coupled to existing media player/recorder applications. Alternatively, CCM 9-300 can be installed as a stand alone mechanism within a client computer system 9-210. Additionally, CCM 9-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD (secure digital card), and/or as part of an installation package. In another example, CCM 9-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a

music CD, reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 9-300 may be installed on client system 9-210 in a clandestine manner, relative to a user.

There are currently two types of copyright licenses recognized by the digital millennium copyright act (DMCA) for the protection of broadcasted copyrighted material. One of the broadcast copyright licenses is a compulsory license, also referred to as a statutory license. A statutory license is defined as a non-interactive license, meaning the user cannot select the song. Further, a caveat of this type of broadcast license is that a user must not be able to select a particular music file for the purpose of recording it to the user's computer system or other storage device. Another caveat of a statutory license is that a media file is not available more than once for a given period of time. In one example, the period of time can be three hours.

The other type of broadcast license recognized by the DMCA is an interactive licensing agreement. An interactive licensing agreement is commonly with the copyright holder, e.g., a record company, the artist, where the copyright holder grants permission for a server, e.g., web server 9-250 and/or content server 9-251 of Figure 9-2 to broadcast copyrighted material. Under an interactive licensing agreement, there are a variety of ways that copyrighted material, e.g., music files, can be broadcast. For example, one manner in which music files can be broadcast is to allow the user to select and listen to a particular sound recording, but without the user enabled to make a sound recording. This is commonly referred to as an interactive with "no save" license, meaning that the end user is unable to save or

store the media content file in a relatively permanent manner. Additionally, another manner in which music files can be broadcast is to allow a user to not only select and listen to a particular music file, but additionally allow the user to save that particular music file to disc and/or burn the music file to CD, MP3 player, or other portable electronic device. This is commonly referred to as an interactive with "save" license, meaning that the end user is enabled to save, store, or burn to CD, the media content file.

It is noted that the DMCA allows for the "perfect" reproduction of the sound recording. A perfect copy of a sound recording is a one-to-one mapping of the original sound recording into a digitized form, such that the perfect copy is virtually indistinguishable and/or has no audible differences from the original recording...

In one embodiment, CCM (copyright compliance mechanism) 9-300 can be stored in web server 9-250 and/or content server 9-251 of network 9-200 and is configured to be installed into each client computer system, e.g., 9-210, 9-220 and 9-230, enabled to access the media files stored within content server 9-251 and/or web server 9-250. Alternatively, copyright compliance mechanism 9-300 can be externally disposed and communicatively coupled with a client computer system 9-200 via, e.g., a portable media device 9-100a of Figure 9-1. In yet another embodiment, CCM 9-300 can be configured to be operable from a media storage device upon which media files may be disposed.

Copyright compliance mechanism 9-300 is configured to be operable while having portions of components, entire components, combinations of components,

disposed within one or more memory units and/or data storage devices of a computer system, e.g., 9-210, 9-220, and/or 9-230.

Additionally, portions of components, entire components and/or combinations of components of CCM 9-300 can be readily updated, e.g., via Internet 9-201, to reflect changes or developments in the DMCA, changes or developments in copyright restrictions and/or licensing agreements that pertain to any media file, changes in current media player applications and/or the development of new media player applications, or to counteract subversive and/or hacker-like attempts to unlawfully obtain one or more media files.

Referring to Figure 9-3, in one embodiment, CCM 9-300 is shown to include instructions 9-301 for enabling client computer system 9-210 to interact with web server 9-250 and content server 9-251 of network 9-200. Instructions 9-301 enable client computer system 9-210 to interact with servers, e.g., 9-250 and 9-251 in a network, e.g., 9-200.

The copyright compliance mechanism 9-300 also includes, in one embodiment, a user ID generator 9-302, for generating a user ID or user key, and one or more cookie(s) which contain(s) information specific to the user and the user's computer system, e.g., 9-210. In one embodiment, the user ID and the cookie(s) are installed in computer system 9-210 prior to installation of the remaining components of the copyright compliance mechanism 9-300. It is noted that the presence of a valid cookie(s) and a valid user ID/user key are verified by web server 9-250 before the remaining components of a CCM 9-300 can be installed, within one

embodiment of the present invention. Additionally, the user ID/user key can contain, but is not limited to, the user's name, the user's address, the user's credit card number, an online payment account number, a verified email address, and an identity (username) and password selected by the user.

Furthermore, the cookie can contain, but is not limited to, information specific to the user, information regarding the user's computer system 9-210, e.g., types of media applications operational therewithin, a unique identifier associated with computer system 9-210, e.g., a MAC (machine address code) address, an IP address, and/or the serial number of the central processing unit (CPU) operable on computer system 9-210 and other information specific to the user and the computer system operated by the user.

Additionally, in another embodiment, user biometrics may be combined with computer system 9-210 data and user data and incorporated into the generation of a user ID. Alternatively, biometric data may be used in a stand alone implementation in the generation of the user ID. Types of biometric data that may be utilized to provide a user ID and/or authorization may include, but is not limited to, fingerprint data, retinal scan data, handprint data, facial recognition data, and the like.

It is noted that the information regarding the client computer system, e.g., 9-210, the user of system 9-210, and an access key described herein can be collectively referred to as authorization data.

Advantageously, with information regarding the user and the user's computer system, e.g., 9-210, web server 9-250 can determine when a user of one computer system, e.g., 9-210, has given their username and password to another user using another computer system, e.g., 9-220. Because the username, password, and the user's computer system 9-210 are closely associated, web server 9-250 can prevent unauthorized access to copyrighted media content, in one embodiment. It is noted that if web server 9-250 detects unauthorized sharing of usernames and passwords, it can block the user of computer system 9-210, as well as other users who unlawfully obtained the username and password, from future access to copyrighted media content available through web server 9-250. Web server 9-250 can invoke blocking for any specified period of time, e.g., for a matter of minutes or hours to months, years, or longer.

Still referring to Figure 9-3, copyright compliance mechanism 9-300 further includes one or more coder/decoders (codec) 9-303 that, in one embodiment, is/are adapted to perform, but is/are not limited to, encoding/decoding of media files, compressing/decompressing of media files, detecting that delivered media files are encrypted as prescribed by CCM 9-300. In the present embodiment, coder/decoder 9-303 can also extract key fields from a header attached to each media content file for, in part, verification that the file originated from a content server, e.g., 9-251.

In the present embodiment, coder/decoder 9-303 can also perform a periodic and repeated check of the media file, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis, to ensure that CCM 9-300 rules are being enforced at any particular moment during

media playback. It is noted that differing coder/decoders 9-303 can be utilized in conjunction with various types of copyrighted media content including, but not limited to, audio files, video files, graphical files, alphanumeric files and the like, such that any type of media content file can be protected in accordance with embodiments of the present invention.

With reference still to Figure 9-3, copyright compliance mechanism 9-300 also includes one or more agent programs 9-304 which are configured to engage in dialogs and negotiate and coordinate transfer of information between a computer system, e.g., 9-210, 9-220, or 9-230, a server, e.g., web server 9-250 and/or content server 9-251, and/or media player applications, with or without recording functionality, that are operable within a client computer system, in one embodiment. In the present embodiment, agent program 9-304 can also be configured to maintain system state, verify that other components are being utilized simultaneously, to be autonomously functional without knowledge of the client, and can also present messages, e.g., error messages, media information, advertising, etc., via a display window or electronic mail. This enables detection of proper skin implementation and detection of those applications that are running. It is noted that agent programs are well known in the art and can be implemented in a variety of ways in accordance with the present embodiment.

Copyright compliance mechanism 9-300 also includes one or more system hooks 9-305, in one embodiment of the present invention. A system hook 9-305 is, in one embodiment, a library that is installed in a computer system, e.g., 9-210, and intercepts system wide events. For example, a system hook 9-305, in conjunction

with skins 9-306, can govern certain properties and/or functionalities of media player applications operating within the client computer system, e.g., 9-210, including, but not limited to, mouse click shortcuts, keyboard shortcuts, standard system accelerators, progress bars, save functions, pause functions, rewind functions, skip track functions, forward track preview, copying to CD, copying to a portable electronic device, and the like.

It is noted that the term govern or governing, for purposes of the present invention, can refer to a disabling, deactivating, enabling, activating, etc., of a property or function. Governing can also refer to an exclusion of that function or property, such that a function or property may be operable but unable to perform in the manner originally intended. For example, during playing of a media file, the progress bar may be selected and moved from one location on the progress line to another without having an effect on the play of the media file.

It is further noted that codec 9-303 compares the information for the media player application operating in client computer system, e.g., 9-210, with a list of "signatures" associated with known media recording applications. In one embodiment, the signature can be, but is not limited to being, a unique identifier of a media player application and which can consist of the window class of the application along with a product name string which is part of the window title for the application. Advantageously, when new media player applications are developed, their signatures can be readily added to the signature list via an update of CCM 9-300 described herein.

The following C++ source code is exemplary implementation of the portion of a codec 9-303 for performing media player application detection, in accordance with an embodiment of the present invention. In another embodiment, the following source code can be modified to detect kernel streaming mechanisms operable within client system 9-210.

```

int
IsRecorderPresent(TCHAR *    szAppClass,
                  TCHAR *    szProdName)
{
    TCHAR    szWndText[_MAX_PATH]; /* buffer to receive title string for
window */
    HWND     hWnd;    /* handle to target window for operation */
    int      nRetVal; /* return value for operation */

    /* initialize variables */
    nRetVal = 0;

    if ( _tcscmp(szAppClass, _T("#32770"))
        == 0)
    {
        /* attempt to locate dialog box with specified window title */
        if ( FindWindow((TCHAR *) 32770, szProdName)
            != (HWND) 0)
        {
            /* indicate application found */
            nRetVal = 1;
        }
    }
    else
    {
        /* attempt to locate window with specified class */
        if ( (hWnd = FindWindow(szAppClass, (LPCTSTR) 0))
            != (HWND) 0)
        {
            /* attempt to retrieve title string for window */
            if ( GetWindowText(hWnd,
                              szWndText,
                              _MAX_PATH)
                != 0)
            {
                /* attempt to locate product name within title string */
                if ( _tcsstr(szWndText, szProdName)

```

```

        != (TCHAR *) 0)
    {
        /* indicate application found */
        nRetVal = 1;
    }
}

/* return to caller */
return nRetVal;
}

```

It is further noted that codec 9-303 can also selectively suppress waveform input/output operations to prevent recording of copyrighted media on a client computer system 9-210. For example, codec 9-303, subsequent to detection of bundled media player applications operational in a client computer system, e.g., 9-210, can stop or disrupt the playing of a media content file. This can be accomplished, in one embodiment, by redirecting and/or diverting certain data pathways that are commonly used for recording, such that the utilized data pathway is governed by the copyright compliance mechanism 9-300. In one embodiment, this can be performed within a driver shim, e.g., wave driver shim 9-309 of Figures 9-5A and 9-5B.

A driver shim can be utilized for nearly any software output device, e.g., a standard Windows™ waveform output device, e.g., Windows™ Media Player, or hardware output device, e.g., speakers or headphones. Client computer system 9-210 is configured such that the driver shim (e.g., 9-309 of Figures 9-5A, 9-5B, 9-5C, and 9-5D) will appear as the default waveform media device to client level application programs. Thus, requests for processing of waveform media input and/or output will pass through the driver shim prior to being forwarded to the actual

waveform audio driver, media device driver 9-505 of Figures 9-5A and 9-5B. Such waveform input/output suppression can be triggered by other components of CCM 9-300, e.g., agent 9-304, to be active when a recording operation is initiated by a client computer system, e.g., 9-210, during the play back of media files which are subject to the DMCA.

It is noted that alternative driver shims can be implemented for nearly any waveform output device including, but not limited to, a Windows™ Media Player. It is further noted that the driver shim can be implemented for nearly any media in nearly any format including, but not limited to, audio media files and audio input and output devices, video, graphic and/or alphanumeric media files and video input and output devices.

The following C++ source code is an exemplary implementation of a portion of a codec 9-303 and/or a custom media device driver 9-307 for diverting and/or redirecting certain data pathways that are commonly used for recording of media content, in accordance with an embodiment of the present invention.

```

DWORD
_stdcall
widMessage(UINT          uDevId,
            UINT          uMsg,
            DWORD         dwUser,
            DWORD         dwParam1,
            DWORD         dwParam2)
{
    BOOL      bSkip;          /* flag indicating operation to be skipped */
    HWND      hWndMon;        /* handle to main window for monitor */
    DWORD     dwRetVal;        /* return value for operation */

    /* initialize variables */
    bSkip = FALSE;

```

```

dwRetVal = (DWORD) MMSYSERR_NOTSUPPORTED;

if (uMsg == WIDM_START)
{
    /* attempt to locate window for monitor application */
    if ( (hWndMon = FindMonitorWindow())
        != (HWND) 0)
    {
        /* obtain setting for driver */
        bDrvEnabled = ( SendMessages(hWndMon,
                                     uiRegMsg,
                                     0,
                                     0)
                      == 0)
                      ? FALSE : TRUE;
    }

    if (bDrvEnabled == TRUE)
    {
        /* indicate error in operation */
        dwRetVal = MMSYSERR_NOMEM;

        /* indicate operation to be skipped */
        bSkip = TRUE;
    }
}

if (bSkip == FALSE)
{
    /* invoke entry point for original driver */
    dwRetVal = CallWidMessage(uDevId, uMsg, dwUser, dwParam1,
dwParam2);
}

/* return to caller */
return dwRetVal;
}

```

It is noted that when properly configured, system hook 9-305 can govern nearly any function or property within nearly any media player application that may be operational within a client computer system, e.g., 9-210 to 9-230. In one embodiment, system hook 9-305 is a DLL (dynamic link library) file. It is further noted that system hooks are well known in the art, and are a standard facility in a

Microsoft Windows™ operating environment, and accordingly can be implemented in a variety of ways. However, it is also noted that system hook 9-305 can be readily adapted for implementation in alternative operating systems, e.g., Apple™ operating systems, Sun Solaris™ operating systems, Linux operating systems, and nearly any other operating system.

In Figure 9-3, copyright compliance mechanism 9-300 also includes one or more skins 9-306, which can be designed to be installed in a client computer system, e.g., 9-210 to 9-230. In one embodiment, skins 9-306 are utilized to assist in client side compliance with the DMCA (digital millennium copyright act) regarding copyrighted media content. Skins 9-306 are customizable interfaces that, in one embodiment, are displayed on a display device (e.g., 9-105) of computer system 9-210 and provide functionalities for user interaction of delivered media content. Additionally, skins 9-306 can also provide a display of information relative to the media content file including, but not limited to, song title, artist name, album title, artist bio, and other features such as purchase inquiries, advertising, and the like.

Furthermore, when system hook 9-305 is unable to govern a function of the media player application operable on a client computer system, e.g., 9-210, such that client computer system could be in non-compliance with DMCA and/or RIAA restrictions, a skin 9-306 can be implemented to provide compliance.

Differing skins 9-306 can be implemented depending upon the DMCA and/or RIAA restrictions applicable to each media content file. For example, in one embodiment, a skin 9-306a may be configured for utilization with a media content file

protected under a non-interactive agreement (DMCA), such that skin 9-306a may not include a pause function, a stop function, a selector function, and/or a save function, etc. Another skin, e.g., skin 9-306b may, in one embodiment, be configured to be utilized with a media content file protected under an interactive with “no save” agreement (DMCA), such that skin 9-306b may include a pause function, a stop function, a selector function, and for those media files having an interactive with “save” agreement, a save or a burn to CD function.

Still referring to Figure 9-3, it is further noted that in the present embodiment, each skin 9-306 can have a unique name and signature. In one embodiment, skin 9-306 can be implemented, in part, through the utilization of an MD (message digest) 5 hash table or similar algorithm. An MD5 hash table can, in one implementation, be a check-sum algorithm. It is well known in the art that a skin, e.g., skin 9-306, can be renamed and/or modified to incorporate additional features and/or functionalities in an unauthorized manner. Since modification of the skin would change the check sum and/or MD5 hash, without knowledge of the MD5 hash table, changing the name or modification of the skin may simply serve to disable the skin, in accordance with one embodiment of the present invention. Since copyright compliance mechanism 9-300 verifies skin 9-306, MD5 hash tables advantageously provide a deterrent against modifications made to the skin.

In one embodiment, copyright compliance mechanism 9-300 also includes one or more custom media device driver(s) 9-307 for providing an even greater measure of control over the media stream while increasing compliance reliability. A client computer system, e.g., 9-210, can be configured to utilize a custom media

device application, e.g., custom media device 9-310 (shown in Figure 9-5B, 9-5C, and 9-5D), to control unauthorized recording of media content files. A custom media device application can be, but is not limited to, a custom media audio device application for media files having sound content, a custom video device application for media files having graphical and/or alphanumeric content, etc. In one embodiment, custom media device 9-310 of Figure 9-5B is an emulation of the custom media device driver 9-307. With reference to audio media, the emulation is performed in a waveform audio driver associated with custom media device 9-310. Driver 9-307 is configured to receive a media file being outputted by system 9-210 prior to the media file being sent to a media output device, e.g., media output device 9-570, and/or a media output application, e.g., recording application 9-502. Examples of a media output device includes, but is not limited to, a video card for video files, a sound card for audio files, etc. Examples of a recording application can include, but is not limited to, CD burner applications for writing to another CDs, ripper applications which capture the media file and change the format of the media file, e.g., from a CD audio file to an .mpeg audio file, and/or a .wav file, and/or an ogg vorbis file, and various other media formats. In one embodiment, client computer system 9-210 is configured with a custom media device driver 9-307 emulating custom media device 9-310, and which is system 9-210's default device driver for media file output. In one embodiment, an existing GUI (graphical user interface) can be utilized or a GUI can be provided, e.g., by utilization of skin 9-306 or a custom web based player application or as part of a CCM 9-300 installation bundle, for forcing or requiring system 9-210 to have driver 9-307 as the default driver.

Therefore, when a media content file is received by system 9-210 from server 9-251, the media content file is playable, provided the media content file passes through the custom media device application (e.g., 9-310 of Figure 9-5B), emulated by custom media device driver 9-307, prior to being outputted. However, if an alternative media player application is selected, delivered media files from server 9-251 will not play on system 9-210.

Thus, secured media player applications would issue a media request to the driver, e.g., 9-307, for the custom media device 9-310 which then performs necessary media input suppression, e.g., waveform suppression for audio files, prior to forwarding the request to the default Windows™ media driver, e.g., waveform audio driver for audio files.

It is noted that requests for non-restricted media files can pass directly through custom media device driver 9-307 to a Windows™ waveform audio driver operable on system 9-210, thus reducing instances of incompatibilities with existing media player applications that utilize waveform media, e.g., audio, video, etc. Additionally, media player applications that do not support secured media would be unaffected. It is further noted that for either secured media or non-restricted media, e.g., audio media files, waveform input suppression can be triggered by other components of CCM 9-300, e.g., agents 9-304, system hooks 9-305, and skins 9-306, or a combination thereof, to be active when a recording operation is initiated simultaneously with playback of secured media files, e.g., audio files. Custom device drivers are well known and can be coded and implemented in a variety of

ways including, but limited to, those found at developers network web sites, e.g., a Microsoft™ or alternative OS (operating system) developer web sites.

Advantageously, by virtue of system 9-210 being configured with a custom media device as the default device driver e.g., device 9-310 of Figures 9-5B, 9-5C, and 9-5D, emulated by a custom media device driver 9-307, those media player applications that require their particular device driver to be the default driver, e.g., Total Recorder, etc., are rendered non-functional for secured music. Further advantageous is that an emulated custom media device provides no native support for those media player applications used as a recording mechanism, e.g., DirectSound capture, (direct sound 9-504 of Figures 9-5A, 9-5B, 9-5C, and 9-5D) etc., that are able to bypass user-mode drivers for most media devices. Additionally, by virtue of the media content being sent through device driver 9-307, thus effectively disabling unauthorized saving/recording of media files, in one embodiment, media files that are delivered in a secured delivery system do not have to be encrypted, although, in another embodiment, they still may be encrypted. By virtue of non-encrypted media files utilizing less storage space and network resources than encrypted media files, networks having limited resources can utilize the functionalities of driver 9-307 of CCM 9-300 to provide compliance with copyright restrictions and/or licensing agreements applicable with a media content file without having the processing overhead of encrypted media files.

Figure 9-4 is an illustration of an exemplary system 9-400 for implementing a copyright compliance mechanism in accordance with an embodiment of the present invention. Specifically, system 9-400 illustrates web server 9-250, content server 9-

251, or a combination of web server 9-250 and content server 9-251 installing a copyright compliance mechanism (e.g., 9-300) in a client's computer system (e.g., 9-210) for controlling media file distribution and controlling user access and interaction of copyrighted media files, in one embodiment of the present invention.

Client computer system 9-210 can communicatively couple with a network (e.g., 9-200) to request a media file, a list of available media files, or a play list of audio files, e.g., MP3 files, etc. In response, web server 9-250 determines if the request originates from a registered user authorized to receive media files associated with the request. If the user is not registered with the network, web server 9-250 can initiate a registration process with the requesting client 9-210. Client registration can be accomplished in a variety of ways. For example, web server 9-250 may deliver to a client 9-210 a registration form having various text entry fields into which the user can enter required information. A variety of information can be required from the user by web server 9-250 including, but not limited to, user's name, address, phone number, credit card number, online payment account number, biometric identification (e.g., fingerprint, retinal scan, etc.), verifiable email address, and the like. In addition, registration can, in one embodiment, include a requirement for the user to select a username and password.

Still referring to Figure 9-4, web server 9-250 can, in one embodiment, detect information related to the client's computer system, e.g., 9-210, and store that information in a user/media database 9-450. For example, web server 9-250 can detect a unique identifier of client computer system 9-210. In one embodiment, the unique identifier can be the MAC (machine address code) address of a NIC (network

interface card) of client computer system 9-210 or the MAC address of the network interface adapter integrated on the motherboard of system 9-210. It is understood that a NIC enables a client computer system 9-210 to access web server 9-250 via Internet 9-201. It is well known that each NIC typically has a unique identifying number MAC address. Further, web server 9-250 can, in one embodiment, detect and store (also in database 9-450) information regarding the types(s) of media player application(s), e.g., Windows Media Player™, Real Player™, iTunes player™ (Apple), Live 365™ player, and those media player applications having recording functionality, e.g., Total Recorder, Cool Edit 2000, Sound Forge, Sound Recorder, Super MP3 Recorder, and the like, that are present and operable in client computer system 9-210. In one embodiment, the client information is verified for accuracy and is then stored in a user database (e.g., 9-450) within web server 9-250.

Subsequent to registration completion, creation of the user ID and password, and obtaining information regarding client computer system 9-210, all or part of this information can be installed in client computer system 9-210. In one embodiment, client computer system 9-210 information can be in the form of a cookie. Web server 9-250 then verifies that the user and client computer system 9-210 data is properly installed therein and that their integrity has not been compromised. Subsequently, web server 9-250 installs a copyright compliance mechanism (e.g., 9-300) into the client's computer system, e.g., 9-210, in one embodiment of the present invention. It is noted that web server 9-250 may not initiate installation of CCM 9-300 until the user ID, password, and client computer system 9-210 information is verified. A variety of common techniques can be employed to install an entire CCM 9-300, portions of components, entire components, and/or combinations or a

function of components. For example, copyright compliance mechanism 9-300 can be installed in a hidden directory within client computer system 9-210, thereby preventing unauthorized access to it. In one embodiment of the present invention, it is noted that unless CCM 9-300 is installed in client computer system 9-210, its user will not be able to request, access, or have delivered thereto, media files stored by web server 9-250 and/or content server 9-251.

Referring still to Figure 9-4, upon completion of client registration and installation of CCM 9-300, client computer system 9-210 can then request a media play list or a plurality of play lists, etc. In response, web server 9-250 determines whether the user of client computer system 9-210 is authorized to receive the media play list associated with the request. In one embodiment, web server 9-250 can request the username and password. Alternatively, web server 9-250 can utilize user database 9-450 to verify that computer 9-210 is authorized to receive a media play list. If client computer 9-210 is not authorized, web server 9-250 can initiate client registration, as described herein. Additionally, web server 9-250 can disconnect computer 9-210 or redirect it to an alternative web site. Regardless, if the user and client computer system 9-210 are not authorized, web server 9-250 will not provide the requested play list to client computer system 9-210.

However, if client computer system 9-210 is authorized, web server 9-210 can check copyright compliance mechanism 9-300 within data base 9-450 to determine if it, or any of the components therein, have been updated since the last time client computer system 9-210 logged in to web server 9-250. If a component of CCM 9-300 has been updated, web server 9-250 can install the updated component and/or

a more current version of CCM 9-300 into client computer system 9-210, e.g., via Internet 9-201. If CCM 9-300 has not been updated, web server 9-250 can then deliver the requested media play list to system 9-210 via Internet 9-201 along with an appended user key or user identification (ID). It is noted that user database 9-450 can also include data for one or more media play lists that can be utilized to provide a media play list to client computer system 9-210. Subsequently, the user of client computer system 9-210 can utilize the received media play list in combination with the media player application operating on system 9-210 to transmit a delivery request for one or more desired pieces of media content from web server 9-250. It is noted that the delivery request contains the user key for validation purposes.

Still referring to Figure 9-4, upon receiving the media content delivery request, web server 9-250 can then check the validity of the requesting media application and the attached user key. In one embodiment, web server 9-250 can utilize user database 9-450 to check their validity. If either or both are invalid, web server 9-250, in one embodiment, can redirect unauthorized client computer system 9-210 to an alternative destination to prevent abuse of the system. However, if both the requesting media application and the user key are valid, CCM 9-300 verifies that skins 9-306 are installed in client computer system 9-210. Additionally, CCM 9-300 further verifies that system hook(s) 9-305 have been run or are running to govern certain functions of those media player applications operable within client computer system 9-210 that are known to provide non-compliance with the DMCA and/or the RIAA. Additionally, CCM 9-300 further diverts and/or redirects certain pathways that are commonly used for recording, e.g., driver 9-307 of Figure 9-5A, device 9-310 of Figure 9-5B, device 9-570 of Figure 9-5C, and driver 9-505 of Figure 9-5D. Once

CCM 9-300 has performed the above described functions, web server 9-250 then, in one embodiment, issues to the client computer 9-210 a redirect command to the current address location of the desired media file content along with an optional time sensitive access key, e.g., for that hour, day, or other defined timeframe.

In response to the client computer system 9-210 receiving the redirect command from web server 9-250, the media player application operating on client computer system 9-210 automatically transmits a new request and the time sensitive access key to content server 9-251 for delivery of one or more desired pieces of media content. The validity of the time sensitive access key is checked by content server 9-251. If invalid, unauthorized client computer 9-210 is redirected by content server 9-250 to protect against abuse of the system and unauthorized access to content server 9-251. If the time sensitive access key is valid, content server 9-251 retrieves the desired media content from content database 9-451 and delivers it to client computer system 9-210. It is noted that, in one embodiment, the delivered media content can be stored in hidden directories and/or custom file systems that may be hidden within client computer system 9-210 thereby preventing future unauthorized distribution. In one embodiment, an HTTP (hypertext transfer protocol) file delivery system is used to deliver the requested media files, meaning that the media files are delivered in their entirety to client computer system 9-210, as compared to streaming media which delivers small portions of the media file.

Still referring to Figure 9-4, it is noted that each media file has, in one embodiment, had a header attached therewith prior to delivery of the media file. In one embodiment, the header can contain information relating to the media file, e.g.,

title or media ID, media data such as size, type of data, and the like. The header can also contain a sequence or key that is recognizable to copyright compliance mechanism 9-300 that identifies the media file as originating from a content server 9-251. In one embodiment, the header sequence/key can also contain instructions for invoking the licensing agreements and/or copyright restrictions that are applicable to that particular media file.

Additionally, if licensing agreements or copyright restrictions are changed, developed, or created, or if new media player applications, with or without recording functionality, are developed, CCM 9-300 would have appropriate modifications made to portions of components, entire components, combinations of components, and/or the entire CCM 9-300 to enable continued compliance with licensing agreements and copyright restrictions. Furthermore, subsequent to modification of copyright compliance mechanism 9-300, modified portions of, or the entire updated CCM 9-300 can easily be installed in client computer system 9-210 in a variety of ways. For example, the updated CCM 9-300 can be installed during client interaction with web server 9-250, during user log-in, and/or while client computer system 9-210 is receiving the keyed play list.

Referring still to Figure 9-4, it is further noted that, in one embodiment, the media files and attached headers can be encrypted prior to being stored within content server 9-251. In one embodiment, the media files can be encrypted utilizing randomly generated keys. Alternatively, variable length keys can be utilized for encryption. It is noted that the key to decrypt the encrypted media files can be stored in a database 9-450, content database 9-451 or in some combination of

databases 9-450 and 9-451. It is further noted that the messages being passed back and forth between client computer system 9-210 and web server 9-250 can also be encrypted, thereby protecting the media files and the data being exchanged from unauthorized use or access. There are a variety of encryption mechanisms and programs that can be implemented to encrypt this data including, but not limited to, exclusive OR, shifting with adds, public domain encryption programs such as Blowfish, and non-public domain encryption mechanisms. It is also noted that each media file can be uniquely encrypted, such that if the encryption code is cracked for one media file, it is not applicable to other media files. Alternatively, groups of media files can be similarly encrypted. Furthermore, in another embodiment, the media files may not be encrypted when being delivered to a webcaster known to utilize a proprietary media player application, e.g., custom media device driver 9-307.

Subsequent to media file decryption, the media file may be passed through CCM 9-300, e.g., a coder/decoder 9-303, to a media player application operating on client computer system 9-210, e.g. playback application 9-501 of Figures 9-5A, 9-5B, 9-5C, 9-5D, and 9-6A, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may or may not be required to experience the media content, e.g., skin 9-306 of Figure 9-3. A skin 9-306 may be necessary when CCM 9-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, an industry standard media player can be utilized by client computer system 9-210 to experience the media content. Typically, many media

player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer, coder/decoder 9-303 will repeatedly ensure that CCM 9-300 rules are being enforced at any particular moment during media playback, shown as step 9-650 of Figure 9-6C.

As the media file content is delivered to the media player application, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 9-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 9-300. If the rules are not enforced, e.g., change due to a user opening up a recording application, e.g., Total Recorder or alternative application, the presentation of the media content is, in one embodiment, suspended or halted. In another embodiment, the presentation of the media content can be modified to output the media content non audibly, e.g., silence. In yet another embodiment, the media content may be audible but recording functionality can be disabled, such that the media content cannot be recorded. These presentation stoppages are collectively shown as step 9-651 of Figure 9-6C.

If the rules, in accordance with CCM 9-300, are enforced, the codec/decoder 9-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 9-210. The newly retrieved portion of the media file is then

presented by the client's media player application. While the newly retrieved portion is presented, CCM 9-300 then again checks that the rules are enforced, and retrieves an additional portion of the media file or suspends presentation of the media file if the rules are not being enforced, and these steps are performed repeatedly throughout the playback of the media file, in a loop environment, until the media file's contents have been presented in their entirety. Advantageously, by constant monitoring during playing of media files, CCM 9-300 can detect undesired activities and enforces those rules as defined by CCM 9-300.

Figure 9-5A is an exemplary logic/bit path block diagram 9-500A showing utilization of a wave shim driver, e.g., wave shim driver 9-309 of Figure 9-3, in conjunction with copyright compliance mechanism 9-300, for selectively controlling recording of copyrighted media received by a client computer system, e.g., system 9-210, in one embodiment of the present invention. Copyright compliance mechanism 9-300 is, in one embodiment, installed and operational on client system 9-210 in the manner described herein.

In one embodiment, a copyright compliance mechanism 9-300 is shown as being communicatively coupled with a media playback application 9-501 via connection 9-520. Therefore, CCM 9-300 is enabled to communicate with playback application 9-501. In one embodiment, CCM 9-300 can be integrated into a media playback application. CCM 9-300 is also coupled to and controls a selectable switch 9-311 in wave shim driver 9-309 (as described in Figure 9-3) via connection 9-522. CCM 9-300 is further coupled to and controls a selectable switch 9-511 in direct sound 9-504 via connection 9-521. Depending upon the copyright restrictions and

licensing agreements applicable to an incoming media file, e.g., 9-499, CCM 9-300 controls whether switches 9-311 and 9-511 are open (shown), thus preventing incoming media 9-499 from reaching a media recording application, or closed (not shown) to allow recording of incoming media 9-499.

For example, incoming media 9-499 may originate from a content server, e.g., 9-251, coupled to system 9-210. In another example, incoming media 9-499 may originate from a personal recording/electronic device, e.g., a MP3 player/recorder or similar device, coupled to system 9-210. Alternatively, incoming media 9-499 may originate from a magnetic, optical or alternative media storage device inserted into a media device player coupled to system 9-210, e.g., a CD or DVD inserted into a CD or DVD player, a hard disk in a hot swappable hard drive, an SD (secure digital card) inserted into a SD reader, and the like. In yet another example, incoming media 9-499 may originate from another media player application or media recording application. Incoming media 9-499 may also originate from a satellite radio feed (e.g., XM radio), a personal communication device (e.g., a mobile phone), a cable television radio input (e.g., DMX (digital music express)), a digital distribution and/or a public presentation source via a network, Internet or other communication connection, pay-per-view and/or pay-per-play system, or a set-top box. It is noted that incoming media 9-499 can originate from nearly any source that can be coupled to system 9-210. However, regardless of the source of incoming media 9-499, embodiments of the present invention, described herein, can prevent unauthorized recording of the media.

Figure 9-5A shows a media playback application 9-501, e.g., an audio, video, or other media player application, operable within system 9-210 and configured to receive incoming media 9-499. Playback application 9-501 can be a playback application provided by an operating system, e.g., Media Player for Windows™ by Microsoft, a freely distributed playback application downloadable from the Internet, e.g., RealPlayer or LiquidAudio, a playback application provided by a webcaster, e.g., PressPlay, or a playback application commercially available.

Figure 9-5A shows media device driver 9-505 which, in one implementation, may be a software driver for a sound card coupled to system 9-210 having a media output device 9-570, e.g., speakers or headphones, coupled therewith for media files having audio content. In another implementation, media device driver 9-505 may be a software driver for a video card coupled with a display device, e.g., 9-105, for displaying media files having alphanumeric and/or graphical content, and so on. With reference to audio files, it is well known that a majority of recording applications assume a computer system, e.g., 9-210, has a sound card disposed therein, providing full-duplex sound functionality to system 9-210. This means media output driver 9-505 can simultaneously cause playback and recording of incoming media files 9-499. For example, media device driver 9-505 can playback media 9-499 along wave-out line 9-539 to media output device 9-570 (e.g., speakers for audible playback) via wave-out line 9-580 while outputting media 9-499 on wave-out line 9-540 to eventually reach recording application 9-502.

For purposes of Figures 9-5A, 9-5B, 9-5C, and 9-5D, the terms wave-in line and wave-out line are referenced from the perspective of media device driver 9-505.

Additionally, for the most part, wave-in lines are downwardly depicted and wave-out lines are upwardly depicted in Figures 9-5A, 9-5B, 9-5C, and 9-5D.

Continuing with Figure 9-5A, playback application 9-501 is coupled with an operating system (O/S) multimedia subsystem 9-503 and direct sound 9-504 via wave-in lines 9-531 and 9-551 respectively. O/S multimedia subsystem 9-503 is coupled to a wave shim driver 9-309 via wave-in line 9-533 and wave-out line 9-546. O/S multimedia subsystem 9-503 is also coupled to a recording application 9-502 via wave-out line 9-548. Operating system (O/S) multimedia subsystem 9-503 can be any O/S multimedia subsystem, e.g., a Windows™ multimedia subsystem for system 9-210 operating under a Microsoft O/S, a QuickTime™ multimedia subsystem for system 9-210 operating under an Apple O/S, and so on. Playback application 9-501 is also coupled with direct sound 9-504 via wave-in line 9-551.

Direct sound 9-504, in one instance, may represent access to a hardware acceleration feature in a standard audio device, enabling lower level access to components within media device driver 9-505. In another instance, direct sound 9-504 may represent a path that can be used by a recording application, e.g., Total Recorder, that can be further configured to bypass the default device driver, e.g., media device driver 9-505 to capture incoming media 9-499 for recording. For example, direct sound 9-504 can be enabled to capture incoming media 9-499 via wave-in line 9-551 and unlawfully output media 9-499 to a recording application 9-502 via wave-out line 9-568, as well as media 9-499 eventually going to media device driver 9-505, the standard default driver.

Still referring to Figure 9-5A, wave shim driver 9-309 is coupled with media device driver 9-505 via wave-in line 9-537 and wave-out line 9-542. Media device driver 9-505 is coupled with direct sound 9-504 via wave-in line 9-553 which is shown to converge with wave-in line 9-537 at media device driver 9-505. Media device driver 9-505 is also coupled with direct sound 9-504 via wave-out line 9-566.

Wave-out lines 9-542 and 9-566 are shown to diverge from wave-out line 9-540 at media device driver 9-505 into separate paths. Wave-out line 9-542 feeds into wave shim driver 9-309 and wave-out line 9-566 feeds into direct sound 9-504. When selectable switch 9-311 and 9-511 are open (shown), incoming media 9-499 cannot flow to recording application 9-502, thus preventing unauthorized recording of it.

For example, incoming media 9-499 is received at playback application 9-501. Playback application 9-501 activates and communicates to CCM 9-300 regarding copyright restrictions and/or licensing agreements applicable to incoming media 9-499. If recording restrictions apply to media 9-499, CCM 9-300 can, in one embodiment, open switches 9-311 and 9-511, thereby blocking access to recording application 9-502, effectively preventing unauthorized recording of media 9-499. In one embodiment, CCM 9-300 can detect if system 9-210 is configured with direct sound 9-504 selected as the default driver to capture incoming media 9-499, via wave-in line 9-551, or a recording application is detected and/or a hardware accelerator is active, such that wave driver shim 9-309 can be bypassed by direct sound 9-504. Upon detection, CCM 9-300 can control switch 9-511 such that the output path, wave-out line 9-568, to recording application 9-502 is blocked. It is

further noted that CCM 9-300 can detect media recording applications and devices as described herein, with reference to Figure 9-3.

Alternatively, if media device driver 9-505 is selected as the default driver, incoming media 9-499 is output from playback application 9-501 to O/S multimedia subsystem 9-503 on wave-in line 9-531. From subsystem 9-503, media 9-499 is output to wave shim driver 9-309 via wave-in line 9-533. The wave shim driver 9-309 was described herein with reference to Figure 9-3. Media 9-499 is output from wave shim driver 9-309 to media device driver 9-505 via wave-in line 9-537. Once received by media device driver 9-505, media 9-499 can be output via wave-out line 9-539 to a media output device 9-570 coupled therewith via wave-out line 9-580. Additionally, media device driver 9-505 can simultaneously output media 9-499 on wave-out line 9-540 back to wave shim driver 9-309. Dependent upon recording restrictions applicable to media 9-499, CCM 9-300 can, in one embodiment, close switch 9-311 (not shown as closed), thereby allowing media 9-499 to be output from wave shim driver 9-309 to subsystem 9-503 (via wave-out line 9-546) and then to recording application 9-502 via wave-out line 9-548. Alternatively, CCM 9-300 can also open switch 9-311, thereby preventing media 9-499 from reaching recording application 9-502.

It is particularly noted that by virtue of CCM 9-300 controlling both switches 9-311 and 9-511, and therefore controlling wave-out line 9-548 and wave-out line 9-568 leading into recording application 9-502, incoming media files, e.g., media 9-499, can be prevented from being recorded in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements

related to the incoming media. It is also noted that embodiments of the present invention in no way interfere with or inhibit the playback of incoming media 9-499.

Figure 9-5B is an exemplary logic/bit path block diagram 9-500B of a client computer system, e.g., 9-210, configured with a copyright compliance mechanism 9-300 for preventing unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 9-300 is, in one embodiment, coupled with and operational on client system 9-210 in the manner with reference to Figures 9-4, 9-5A, 9-5C, 9-5D, 9-6, and 9-7.

Diagram 9-500B of Figure 9-5B is similar to diagram 9-500A of Figure 9-5A, with a few changes. Particularly, diagram 9-500B includes a custom media device 9-310 communicatively interposed between and coupled to O/S multimedia subsystem 9-503 and wave shim driver 9-309. Custom media device 9-310 is coupled to O/S multimedia subsystem via wave-in line 9-533 and wave-out line 9-546. Custom media device 9-310 is coupled with wave shim driver 9-309 via wave-in line 9-535 and wave-out line 9-544. Additionally, custom media device 9-310 is coupled with direct sound 9-504 via wave-in line 9-553 which converges with wave-in line 9-533 and wave-out line 9-566 which diverges from wave-out line 9-546, in one embodiment.

Also added to Figure 9-5B is a media hardware output device 9-570 that is coupled to media device hardware driver 9-505 via line 9-580. Media hardware output device 9-570 can be, but is not limited to, a sound card for audio playback, a video card for video, graphical, alphanumeric, etc, output, and the like.

In one embodiment, CCM 9-300 is communicatively coupled with playback application 9-501 via connection 9-520, waveform driver shim 9-309 via connection 9-522, and custom media device 9-310, via connection 9-521. CCM 9-300 is coupled to and controls a selectable switch 9-311 in waveform driver shim 9-309 via connection 9-522. CCM 9-300 is also coupled to and controls a selectable switch 9-312 in custom audio device 9-310 via connection 9-521. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 9-499, CCM 9-300 controls whether switches 9-311 and 9-312 are open (shown), thus preventing the incoming media 9-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 9-499.

Continuing with Figure 9-5B, direct sound 9-504 is shown coupled with custom media device 9-310 via wave-in line 9-553, instead of being coupled with media device driver 9-505 (Figure 9-5A). In one embodiment, custom audio device 9-310 mandates explicit selection through system 9-210, meaning that custom audio device 9-310 needs to be selected as a default driver of system 9-210. By virtue of having the selection of custom media device 9-310 as the default driver of system 9-210, the data path necessary for direct sound 9-504 to capture the media content is selectively closed.

For example, incoming media 9-499 originating from nearly any source with reference to Figure 9-5A is received by media playback application 9-501 of system 9-210. Playback application 9-501 communicates to CCM 9-300, via connection 9-

520, to determine whether incoming media 9-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 9-501 communicates with CCM 9-300 to control switch 9-311 and 9-312 accordingly. In the present example, recording of incoming media 9-499 would violate applicable restrictions and/or agreements and therefore switch 9-312 is in an open position, such that the output path to recording application 9-502, e.g., wave-out line 9-548 and/or wave-out line 9-568, is effectively blocked, thereby preventing unauthorized recording of media 9-499.

Alternatively, if media device driver 9-505 is selected as the default driver, incoming media 9-499 continues from O/S multimedia subsystem 9-503, through custom audio device 9-310, wave driver shim 9-309, and into media device driver 9-505 where media 9-499 can be simultaneously output to media output device 9-570 via line 9-580, and output on wave-out line 9-540 to wave-and outputted by media device driver 9-505 to wave shim driver 9-309 on wave-out line 9-542. However, by virtue of CCM 9-300 controlling switch 9-311, wave-out line 9-544 which eventually leads to recording application 9-502 is blocked, thus effectively preventing unauthorized recording of media 9-499.

It is particularly noted that by virtue of CCM 9-300 controlling both switches 9-311 and 9-312 and therefore controlling wave-out line 9-548 and wave-out line 9-568, any incoming media files, e.g., incoming media 9-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 9-5B, it is further noted that custom media device 9-310 allows for unfettered playback of incoming media 9-499. Additionally, at any time during playback of media 9-499, custom media device 9-310 can be dynamically activated by CCM 9-300.

Figure 9-5C is an exemplary logic/bit path block diagram 9-500C of a client computer system, e.g., 9-210, configured with a copyright compliance mechanism 9-300 for preventing unauthorized output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 9-300 is, in one embodiment, coupled with and operational on client system 9-210 in the manner with reference to Figures 9-4, 9-5A, 9-5B, 9-5D, 9-6, and 9-7.

Diagram 9-500C of Figure 9-5C is similar to diagram 9-500B of Figure 9-5B, with a few changes. Particularly, diagram 9-500C includes a media hardware output device 9-570 that is coupled with a media device driver 9-505. In one embodiment, media hardware output device 9-570 can be a S/PDIF (Sony/Phillips Digital Interface) card for providing multiple outputs, e.g., an analog output 9-573 and a digital output 9-575. An alternative media hardware output device providing similar digital output can also be implemented as device 9-570 including, but not limited to, a USB (universal serial bus) output device and/or an externally accessible USB port located on system 9-210, a FireWire (IEEE1394) output device and/or an externally accessible FireWire port located on system 9-210, with wireline or wireless functionality. In the present embodiment, media hardware output device 9-570 is shown to include a switch 9-571 controlled by CCM 9-300 via communication line 9-

523, similar to switches 9-311 and 9-312, for controlling output of incoming media 9-499.

In one embodiment, CCM 9-300 is communicatively coupled with playback application 9-501 via connection 9-520, waveform driver shim 9-309 via connection 9-522, custom media device 9-310, via connection 9-521, and media hardware output device 9-570 via connection 9-523. CCM 9-300 is coupled to and controls a selectable switch 9-311 in waveform driver shim 9-309 via connection 9-522. CCM 9-300 is also coupled to and controls a selectable switch 9-312 in custom audio device 9-310 via connection 9-521. CCM 9-300 is further coupled to and controls a selectable switch 9-571 in media hardware output device 9-570 via connection 9-523. Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 9-499, CCM 9-300 controls whether switches 9-311 and 9-312 are open (shown), thus preventing the incoming media 9-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 9-499. Additionally, CCM 9-300 controls whether switch 9-571 is open (shown), thus preventing incoming media 9-499 from being output from digital output 9-575 of media hardware output device 9-570, or closed (not shown) to allow incoming media 9-499 to be output from media hardware output device 9-570.

By controlling media hardware output device 9-570, copyright compliance mechanism 9-300 can prevent unauthorized output of incoming media 9-499 to, e.g., a digital recording device that may be coupled with digital output 9-575 of media hardware output device 9-570. Accordingly, in one embodiment, CCM 9-300 is enabled to also detect digital recording devices that may be coupled to a digital

output line, e.g., 9-571, of a media hardware output device, e.g., 9-570. Examples of a digital recording device that can be coupled to media hardware output device 9-570 can include, but is not limited to, mini-disc recorders, MP3 recorders, personal digital recorders, digital recording devices coupled with multimedia systems, personal communication devices, set-top boxes, and/or nearly any digital device that can capture an incoming media 9-499 being output from a media hardware output device 9-570, e.g., a sound card.

Continuing with Figure 9-5C, direct sound 9-504 is shown coupled with custom media device 9-310 via wave-in line 9-553, instead of being coupled with media device driver 9-505 (Figure 9-5A). In one embodiment, custom audio device 9-310 mandates explicit selection through system 9-210, meaning that custom audio device 9-310 is needs to be selected as a default driver of system 9-210. By virtue of having the selection of custom media device 9-310 as the default driver of system 9-210, the data path necessary for direct sound 9-504 to capture the media content is selectively closed.

For example, incoming media 9-499 originating from nearly any source with reference to Figure 9-5A is received by media playback application 9-501 of system 9-210. Playback application 9-501 communicates to CCM 9-300, via connection 9-520, to determine whether incoming media 9-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 9-501 communicates with CCM 9-300 to control switch 9-311, 9-312, and 9-571 accordingly. In the present example, recording of incoming media 9-499 would violate applicable restrictions and/or agreements and therefore switch 9-312 is in an open position,

such that the output path to recording application 9-502, e.g., wave-out line 9-548 and/or wave-out line 9-568, is effectively blocked, thereby preventing unauthorized recording of media 9-499.

Alternatively, if media device driver 9-505 is selected as the default driver, incoming media 9-499 continues from O/S multimedia subsystem 9-503, through custom audio device 9-310, wave driver shim 9-309, and into media device driver 9-505 where media 9-499 can be simultaneously output to media output device 9-570 via line 9-580, and output on wave-out line 9-540 to wave-and outputted by media device driver 9-505 to wave shim driver 9-309 on wave-out line 9-542. However, by virtue of CCM 9-300 controlling switch 9-311, wave-out line 9-544 which eventually leads to recording application 9-502 is blocked, thus effectively preventing unauthorized recording of media 9-499.

It is particularly noted that by virtue of CCM 9-300 controlling both switches 9-311 and 9-312 and therefore controlling wave-out line 9-548 and wave-out line 9-568, any incoming media files, e.g., incoming media 9-499, can be prevented from being recording in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Still referring to Figure 9-5C, it is particularly noted that although CCM 9-300 can prevent unauthorized recording of incoming media 9-499 by controlling switches 9-311 and 9-312, thus preventing incoming media 9-499 from reaching recording application 9-502, controlling switches 9-311 and 9-312 do nothing to prevent incoming media 9-499 from being captured by a peripheral digital device, e.g., a

mini-disc recorder, etc., coupled to a digital output 9-575 of device 9-570. Thus, by also controlling the output, via digital output 9-575 of media hardware output device 9-570, through control of switch 9-571, CCM 9-300 can prevent unauthorized capturing of incoming media 9-499 during output, e.g., on a sound card for audio files, a video card for video and/or graphical files, regardless of whether incoming media 9-499 is received in a secure and encrypted manner. However, when switch 9-571 is in a closed position, incoming media 9-499 may be played back in an unfettered manner. Additionally, at any time during playback of media 9-499, switch 9-312 of custom media device 9-310, switch 9-311 of media device driver 9-309, and/or switch 9-571 of media hardware output device 9-570 can be dynamically activated by CCM 9-300.

Figure 9-5D is an exemplary logic/bit path block diagram 9-500D of a client computer system, e.g., 9-210, configured with a copyright compliance mechanism 9-300 for preventing unauthorized kernel based output and unauthorized recording of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 9-300 is, in one embodiment, coupled with and operational on client system 9-210 in the manner described herein with reference to Figures 9-4, 9-5A, 9-5B, 9-5C, 9-6, and 9-7.

Diagram 9-500D of Figure 9-5D is similar to diagram 9-500C of Figure 9-5C, with some changes. Particularly, diagram 9-500D includes a kernel streaming mechanism 9-515, e.g., DirectKS, that is coupled with a media device driver 9-505. In one embodiment, DirectKS 9-515 can be used for establishing a direct connection with media device driver 9-505. In the present embodiment, media device driver 9-

505 is shown to include a switch 9-511 controlled by CCM 9-300 via communication line 9-524, that is similar to switches 9-311, 9-312, and 9-571, for controlling output of incoming media 9-499.

In one embodiment, CCM 9-300 is communicatively coupled with: playback application 9-501 via connection 9-520, waveform driver shim 9-309 via connection 9-522, custom media device 9-310, via connection 9-521, and media device driver 9-505 via connection 9-524. Specifically, CCM 9-300 is coupled to and controls a selectable switch 9-311 in waveform driver shim 9-309 via connection 9-522. CCM 9-300 is also coupled to and controls a selectable switch 9-312 in custom audio device 9-310 via connection 9-521. CCM 9-300 is further coupled to and controls a selectable switch 9-511 in media device driver 9-505 via connection 9-524.

Depending upon the copyright restrictions and licensing agreements applicable to an incoming media file, e.g., media 9-499, CCM 9-300 controls whether switches 9-311 and 9-312 are open (shown), thus preventing the incoming media 9-499 from reaching a recording application, or closed (not shown) so as to allow recording of the incoming media 9-499. Additionally, CCM 9-300 controls whether switch 9-511 is open (shown), thus preventing incoming media 9-499 from being returned from media device driver 9-505 to playback application 9-501, where DirectKS 9-515 can capture incoming media 9-499 and redirect it to recording application 9-502 to create an unauthorized copy or recording of incoming media 9-499. CCM 9-300 can also control whether switch 9-511 is closed (not shown) to allow incoming media 9-499 to be returned to playback application 9-501, where DirectKS 9-515 can capture and redirect incoming media 9-499 to recording application 9-502.

DirectKS 9-515, in one embodiment, may represent a kernel streaming mechanism that is adapted to establish a direct connection with a media device driver 9-505 of an operating system operable on client computer system 9-210, enabling kernel level access to media device driver 9-505. A kernel streaming mechanism can be implemented for the purpose of precluding utilization of standard audio APIs (application programming interfaces) to play or record media content, with particular attention paid to those playback applications with low latency requirements. DirectKS 9-515 can bypass existing APIs and communicate with media device driver 9-505. DirectKS 9-515 can be readily adapted to work in conjunction with a playback application, e.g., 9-501, to capture and redirect incoming media 9-499 to recording application 9-502, via wave-out line 9-588. Accordingly, DirectKS 9-515 can be implemented to create unauthorized media recordings.

By controlling media device driver 9-505, copyright compliance mechanism 9-300 can prevent unauthorized output of incoming media 9-499 to, e.g., a digital recording device 9-529 that may be coupled with recording application 9-502. In one embodiment, media device driver 9-505 is configured through the kernel mixer (not shown) to control the data path. Additionally, in one embodiment, CCM 9-300 is enabled to also detect a kernel streaming mechanism 9-515 (e.g., DirectKS) that may be operable on client computer system 9-210, as described herein with reference to Figure 9-3.

In one embodiment, custom media device 9-310 mandates explicit selection through system 9-210, meaning that custom media device 9-310 is needs to be selected as a default driver of system 9-210. By virtue of having the selection of

custom media device 9-310 as the default driver of system 9-210, the data path necessary for direct sound 9-504 to capture the media content is selectively closed.

For example, incoming media 9-499 originating from nearly any source with reference to Figure 9-5A is received by media playback application 9-501 of system 9-210. Playback application 9-501 communicates to CCM 9-300, via connection 9-520, to determine whether incoming media 9-499 is protected by any copyright restrictions and/or licensing agreements. Playback application 9-501 communicates with CCM 9-300 to control switches 9-311, 9-312, 9-571, and 9-511, accordingly. In the present example, recording of incoming media 9-499 would violate applicable restrictions and/or agreements and therefore switch 9-511 is in an open position, such that the output path to recording application 9-502, e.g., wave-out line 9-548 and/or wave-out line 9-568 and/or wave-out line 9-588, is effectively blocked, thereby preventing unauthorized recording of media 9-499.

Still referring to Figure 9-5D, it is particularly noted that although CCM 9-300 can prevent unauthorized recording of incoming media 9-499 by controlling switches 9-311, 9-312, and 9-571, thus preventing incoming media 9-499 from reaching recording application 9-502, controlling switches 9-311, 9-312, and 9-571, do nothing to prevent incoming media 9-499 from being returned to recording application 9-502 by a kernel streaming mechanism 9-515(e.g., DirectKS), which enables capturing and redirecting of incoming media 9-499 to recording application 9-502, via wave-out line 9-588. Thus, by also controlling switch 9-511 of media device driver 9-505, CCM 9-300 can prevent kernel streaming mechanism 9-515 from returning incoming media 9-499 to recording application 9-502, thereby preventing incoming media 9-

499 from being captured and redirected to recording application 9-502 in an attempt to create an unauthorized copy and/or recording of incoming media 9-499.

However, when switch 9-511 is in a closed position, incoming media 9-499 may be returned to a recording application 9-502, such that recording could be possible, provided recording does not violate copyright restrictions applicable to incoming media 9-499. Additionally, at any time during playback of media 9-499, switch 9-312 of custom media device 9-310, switch 9-311 of wave shim driver 9-309, and/or switch 9-511 of media device driver 9-505 can be dynamically activated by CCM 9-300.

Figure 9-6A is a block diagram of a media file, e.g., incoming media 9-499, adapted to be received by a playback application, e.g., 9-501 of Figures 9-5A, 9-5B, 9-5C, and 9-5D, configured with an indicator 9-605 for enabling incoming media 9-499 to comply with rules according to the SCMS (serial copy management system). When applicable to a media file, e.g., 9-499, the SCMS allows for one copy of a copyrighted media file to be made, but not for copies of copies to be made. Thus, if incoming media 9-499 can be captured by a recording application, e.g., 9-502 of Figures 9-5A, 9-5B, 9-5C, and/or 9-5D, and/or a recording device, e.g. 9-529, and/or a peripheral recording device and/or a recording application coupled to a digital output of a media hardware output device, e.g., digital output 9-575 of media hardware output device 9-570 of Figures 9-5B, 9-5C, and 9-5D, and/or a kernel streaming mechanism 9-515, e.g., DirectKS of Figure 9-5D, unauthorized copying and/or recording may be accomplished.

Playback application 9-501 is coupled with CCM 9-300 via communication line 9-520 in a manner analogous to Figures 9-5A, 9-5B, 9-5C, and/or 9-5D. Although not shown in Figure 9-6, it is noted that CCM 9-300 is also coupled to switches 9-311 and 9-511 as shown in Figure 9-5A, switches 9-311 and 9-312 in Figure 9-5B, switches 9-311, 9-312, and 9-571 in Figure 9-5C, and switches 9-312, 9-311, 9-571, and 9-511, in Figure 9-5D.

In one embodiment, an indicator 9-605 is attached to incoming media 9-499 for preventing unauthorized copying or recording in accordance with the SCMS. In one embodiment, indicator 9-605 can be a bit that may be transmitted prior to beginning the delivery of incoming media 9-499 to playback application 9-501. In another embodiment, indicator 9-605 may be placed at the beginning of the bit stream of incoming media 9-499. In another embodiment, indicator 9-605 may be placed within a frame period of incoming media 9-499, e.g., every fifth frame, or any other desired frame period. In another embodiment, indicator 9-605 may be transmitted at a particular time interval or intervals during delivery of the media file, e.g. incoming media 9-499. Thus, indicator 9-605 may be placed nearly anywhere within or attached to the bit stream related to incoming media 9-499.

Indicator 9-605 may be comprised of various indicators, e.g., a level 0 indicator, a level 1 indicator, and a level 2 indicator, in one embodiment of the present invention. In the present embodiment, a level 0 indicator may be for indicating to CCM 9-300 that copying is permitted without restriction, e.g., incoming media 9-499 is not copyrighted or that the copyright is not asserted. In the present embodiment, a level 1 indicator may be for indicating to CCM 9-300 that one

generation of copies of incoming media 9-499 may be made, such that incoming media 9-499 is an original copy and that one copy may be made. In the present embodiment, a level 2 indicator may be for indicating to CCM 9-300 that incoming media 9-499 is copyright protected and/or a copy thereof, and as such no digital copying is permitted.

For example, incoming media 9-499 is received by playback application 9-501. Application 9-501 detects an indicator 9-605 attached therewith, in this example, a level 2 bit is placed in the bit stream for indicating to CCM 9-300 that copying is not permitted.

For example, when CCM 9-300 is configured in system 9-210 such as that shown in Figure 9-5A, in response to a level 2 indicator bit, CCM 9-300, while controlling the audio path, then activates switches 9-311 and 9-511 to prevent any recording of incoming media 9-499.

When CCM 9-300 is configured in system 9-210 such as that shown in Figure 9-5B, in response to a level 2 indicator bit, CCM 9-300, while controlling the audio path, then activates switches 9-311 and 9-312 to prevent any recording of incoming media 9-499.

When CCM 9-300 is configured in system 9-210 such as that shown in Figure 9-5C, in response to a level 2 indicator bit, CCM 9-300, while controlling the audio path, then activates switches 9-311, 9-312, and 9-571 to prevent any recording of incoming media 9-499.

It is noted that CCM 9-300 can activate or deactivate switches coupled therewith, as described herein with reference to Figures 9-5A, 9-5B, 9-5C, and 9-5D, thereby funneling incoming media 9-499 through the secure media path, in this instance the audio path, to prevent unauthorized copying of incoming media 9-499. It is further noted that CCM 9-300 can detect media recording applications and devices as described herein, with reference to Figure 9-3.

Figures 9-7A, 9-7B, and 9-7C, are a flowchart 9-700 of steps performed in accordance with one embodiment of the present invention for controlling end user interaction of delivered electronic media. Flowchart 9-700 includes processes of the present invention which, in one embodiment, are carried out by processors and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 9-104 and/or computer usable non-volatile memory 9-103 of Figure 9-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 9-700, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figures 9-7A, 9-7B, and 9-7C. Within the present embodiment, it should be appreciated that the steps of flowchart 9-700 may be performed by software, by hardware or by any combination of software and hardware.

The present embodiment provides a mechanism for restricting recording of high fidelity media content delivered via one or more communication networks. The present embodiment delivers the high fidelity media content to registered clients while preventing unauthorized clients from directly receiving media content from a source database. Once the client computer system receives the media content, it can be stored in hidden directories and/or custom file systems that may be hidden to prevent subsequent unauthorized sharing with others. It is noted that various functionalities can be implemented to protect and monitor the delivered media content. For example, the physical address of the media content can be hidden from media content recipients. In another example, the directory address of the media content can be periodically changed. Additionally, an access key procedure and rate control restrictor can also be implemented to monitor and restrict suspicious media content requests. Furthermore, a copyright compliance mechanism, e.g., CCM 9-300, can be installed in the client computer system 9-210 to provide client side compliance with licensing agreements and copyright restrictions applicable to the media content. By implementing these and other functionalities, the present embodiment restricts access to and the distribution of delivered media content and provides a means for copyrighted media owner compensation.

It is noted that flowchart 9-700 is described in conjunction with Figures 9-2, 9-3, 9-4, 9-5A, 9-5B, 9-5C, and 9-5D, in order to more fully describe the operation of the present embodiment. In step 9-702 of Figure 9-7A, a user of a computer system, e.g., 9-210, causes the computer to communicatively couple to a web server, e.g., 9-250, via one or more communication networks, e.g., Internet 9-201, and proceeds to

attempt to log in. It is understood that the log in process of step 9-702 can be accomplished in a variety of ways in accordance with the present invention.

In step 9-704 of Figure 9-7A, web server 9-250 accesses a user database, e.g., 9-450, to determine whether the user and the computer system 9-210 logging in are registered with it. If the user and computer system 9-210 are registered with web server 9-250, the present embodiment proceeds to step 9-714. However, if the user and computer system 9-210 are logging in for the first time, web server 9-250 can initiate a user and computer system 9-210 registration process at step 9-706.

In step 9-706, registration of the user and computer system 9-210 is initiated. The user and computer system registration process can involve the user of computer system 9-210 providing personal information including, but not limited to, their name, address, phone number, credit card number, online payment account number, biometric identification (e.g., fingerprint, retinal scan, etc.), and the like. Web server 9-250 can verify the accuracy of the information provided. Web server 9-250 can also acquire information regarding the user's computer system 9-210 including, but not limited to, identification of media players disposed and operable on system 9-210, a unique identifier corresponding to the computer system, etc. In one embodiment, the unique identifier corresponding to the computer system can be a MAC address. Additionally, web server 9-250 can further request that the user of computer system 9-210 to select a username and password.

In step 9-708 of Figure 9-7A, subsequent to the completion of the registration process, web server 9-250 generates a unique user identification (ID) or user key

associated with the user of client computer system 9-210. The unique user ID, or user key, is then stored by web server 9-250 in a manner that is associated with that registered user. Furthermore, one or more cookies containing that information specific to that user and the user's computer system 9-210, is installed in a non-volatile memory device, e.g., 9-103 and/or data storage device 9-108 of computer system 9-210. It is noted that the user ID and cookie can be stored in a hidden directory within one or more non-volatile memory devices within computer system 9-210, thereby preventing user access and/or manipulation of that information. It is further noted that if the unique user ID, or user key, has been previously generated for the user and computer 9-210 that initially logged-in at step 9-702, the present embodiment proceeds to step 9-714

In step 9-710, web server 9-250 verifies that the user ID and the cookie(s) are properly installed in computer system 9-210 and verifies the integrity of the cookie(s) and the user ID, thereby ensuring no unauthorized alterations to the user ID or the cookie has occurred. If the user ID is not installed and/or not valid, web server 9-250 can re-initiate the registration process at step 9-706. Alternatively, web server 9-250 can decouple computer system 9-210 from the network, thereby requiring a re-log in by the user of computer 9-210. If the cookie(s) and user ID are valid, the present embodiment proceeds to step 9-712.

In step 9-712 of Figure 9-7A, web server 9-250 can install a version of a copyright compliance mechanism, e.g., 9-300, onto one or more non-volatile memory devices of computer system 9-210. Installing CCM 9-300 into user's computer system 9-210 can facilitate client side compliance with licensing agreements and

copyright restrictions applicable to specific delivered copyrighted media content. At step 9-712, the components of CCM 9-300, such as instructions 9-301, coder/decoder (codec) 9-303, agent programs 9-304, system hooks 9-305, skins 9-306, and custom media device drivers 9-307 (e.g., custom media device 9-310 of Figures 9-5B, 9-5C, and 9-5D), are installed in computer system 9-210, such as that shown in Figures 9-5A, 9-5B, 9-5C, and 9-5D. In one embodiment, a hypertext transfer protocol file delivery system can be utilized to install CCM 9-300 into computer system 9-210. However, step 9-712 is well suited to install CCM 9-300 on computer system 9-210 in a wide variety of ways in accordance with the present embodiment. For example, CCM 9-300 can be installed as an integrated component within a media player application, media recorder application, and/or media player/recorder applications. Alternatively, CCM 9-300 can be installed as a stand alone mechanism within a client computer system 9-210. Additionally, CCM 9-300 can be installed as a stand alone mechanism and/or as part of a bundled application from a media storage device, e.g., a CD, a DVD, an SD, and/or as part of an installation package. In another embodiment, CCM 9-300 can be installed in conjunction with a presentation of desired media content, e.g., listening to an audio file on a music CD, reading a document, viewing a video, etc. It is noted that, in one embodiment, CCM 9-300 may be installed on client system 9-210 in a clandestine manner, relative to a user.

In step 9-714, web server 9-250 can request the previously established username and password of the user of client computer system 9-210. Accordingly, the user of client computer system 9-210 causes it to transmit to web server 9-250 the previously established username and password. Upon the receipt thereof, web

server 9-250 may access a user database, e.g., 9-450, to determine their validity. If the username and password are invalid, web server 9-250 refuses access wherein flowchart 9-500 may be discontinued (not shown). Alternatively, if the username and password are valid, the present embodiment proceeds to step 9-716.

In step 9-716 of Figure 9-7A, web server 9-250 can access media file database 9-450 to determine if copyright compliance mechanism 9-300 has been updated to reflect changes made to the DMCA (digital millennium copyright act) and/or to the interactive/non-interactive licensing agreements recognized by the DMCA. It is noted that alternative licensing agreements can be incorporated into copyright compliance mechanism 9-300. Advantageously, by providing a copyright compliance mechanism that can be readily updated to reflect changes in existing copyright restrictions and/or the introduction of other types of licensing agreements, and/or changes to existing media player applications, or the development of new media player applications, copyright compliance mechanism 9-300 can provide compliance with current copyright restrictions.

Continuing with step 9-716, if web server 9-250 determines that CCM 9-300, or components thereof, of computer 9-210 has been updated, web server 9-250 initiates installation of the newer components and/or the most current version of CCM 9-300 into computer system 9-210, shown as step 9-718. If web server 9-250 determines that the current version of CCM 9-300 installed on system 9-210 does not have to be updated, the present embodiment proceeds to step 9-720 of Figure 9-7B.

In step 9-720 of Figure 9-7B, the user of client computer system 9-210 causes it to transmit to web server 9-250, e.g., via Internet 9-201, a request for a play list of available media files. It is noted that the play list can contain all or part of the media content available from a content server, e.g., 9-251.

In step 9-722, in response to web server 9-250 receiving the play list request, web server 9-250 transmits to client computer system 9-210 a media content play list together with the unique user ID associated with the logged-in user. The user ID, or user key, can be attached to the media content play list in a manner invisible to the user. It is noted that the media content in content server 9-251 can be, but is not limited to, high fidelity music, audio, video, graphics, multimedia, alphanumeric data, and the like. The media content play list of step 9-720 can be implemented in diverse ways. In one example, web server 9-250 can generate a media content play list by combining all the available media content into a single play list. Alternatively, all of the media content titles, or different lists of titles, can be loaded from content server 9-251 and passed to a CGI (common gateway interface) program operating on web server 9-250 where the media titles, or differing lists of titles, can be concatenated into a single dimensioned array that can be provided to client computer system 9-210. It is understood that the CGI can be written in nearly any software computing language.

In step 9-724 of Figure 9-7B, the user of client computer system 9-210 can utilize the received media content play list in conjunction with a media player application in order to cause client computer system 9-210 to transmit a request to web server 9-250 for delivery of desired media content, and wherein the user ID is

automatically included therewith. The media content play list provided to client computer system 9-210 by web server 9-250 can enable the user to create one or more customized play lists by the user selecting desired media content titles. It is noted that a customized media play list can establish the media content that will eventually be delivered to client computer system 9-250 and the order in which the content will be delivered. Additionally, the user of client computer system 9-250 can create one or more customized play lists and store those play lists in system 9-250 and/or within web server 9-250. It is noted that a customized play list does not actually contain the desired media content titles, but rather the play list includes one or more identifiers associated with the desired media content that can include, but is not limited to, a song, an audio clip, a video clip, a picture, a multimedia clip, an alphanumeric document, or particular portions thereof. In another embodiment, the received media content play list can include a random media content delivery choice that the user of client computer system 9-210 can transmit to web server 9-250, with the user ID, to request delivery of the media content in a random manner.

In step 9-726, upon receiving the request for media content from client computer system 9-210, web server 9-250 determines whether the requesting media application operating on client computer system 9-210 is a valid media application. One of the functions of a valid media application is to be a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If web server 9-250 determines that the media application operating on system 9-210 is not a valid media application, the present embodiment proceeds to step 9-727 which in one embodiment, redirects client computer system 9-210 to a web site where the user of system 9-210 can download a valid media

player application or to a software application which can identify client computer system 9-210, log system 9-210 out of web server 9-250 and/or prevent future logging-in for a defined period of time, e.g., 15 minutes, an hour, a day, a week, a month, a year, or any specified amount of time. If web server 9-250 determines that the media application operating on system 9-210 is a valid media application, the present embodiment proceeds to step 9-728.

In step 9-728 of Figure 9-7B, the present embodiment causes web server 9-250 to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer system 9-210 is valid. If web server 9-250 determines that the user ID is invalid, the present embodiment proceeds to step 9-729 where client computer system 9-210 can be logged off web server 9-250 or client computer system 9-250 can be returned to step 9-706 (of Figure 9-7A) to re-register and to have another unique user ID generated by web server 9-250. It is noted that the order in which steps 9-726 and 9-728 are performed can be altered such that step 9-728 can be performed prior to step 9-726. If web server 9-250 determines that the user ID is valid, the present embodiment proceeds to step 9-730.

In step 9-730, prior to web server 9-250 authorizing the delivery of the redirect and access key for the requested media file content, shown as step 9-732, CCM 9-300 governs certain media player applications and/or functions thereof that are operable on client computer system 9-210. These governed functions can include, pause, stop, progress bar, save, etc. It is noted that, in one embodiment, CCM 9-300 can utilize system hooks 9-305 to accomplish the functionality of step 9-730.

In step 9-732 of Figure 9-7C, the present embodiment causes web server 9-250 to transmit to client computer system 9-210 a redirection command along with a time sensitive access key (for that hour, day or for any defined period of time) thereby enabling client computer system 9-210 to receive the requested media content. The redirection command can include a time sensitive address of the media content location within content server 9-251. The address is time sensitive because, in one embodiment, the content server 9-251 periodically renames some or all of the media address directories, thereby making previous content source addresses obsolete. Alternatively, the address of the media content is changed. In another embodiment, the location of the media content can be changed along with the addresses. Regardless, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server 9-251. Therefore, if someone with inappropriate or unlawful intentions is able to find where the media content is stored, subsequent attempts will fail, as the previous route no longer exists, thereby preventing future unauthorized access.

It is noted that in one embodiment of the present invention, the addresses (or routes) of content server 9-251 that are actively coupled to one or more client computer systems (e.g., 9-210 to 9-230) are maintained while future addresses, or routes, are being created for new client devices. It is further noted that as client computer systems are uncoupled from the media content source of content server 9-251, that directory address, or link, can be immediately changed, thereby preventing unauthorized client system or application access.

In another embodiment, the redirection of client computer system 9-210 to content server 9-251 can be implemented by utilizing a server network where multiple servers are content providers, (e.g., 9-251), or by routing a requesting client computer system (e.g., 9-210, 9-220, or 9-230) through multiple servers. In yet another embodiment, the delivery of media content from a central content provider (e.g., 9-251) can be routed through one or more intermediate servers before being received by the requesting client computer system, e.g., 9-210 to 9-230.

The functionality of step 9-732 is additionally well suited to provide recordation of the Internet Protocol (IP) addresses of the client computer systems, e.g., 9-210, the media content requested and its transfer size, thereby enabling accurate monitoring of royalty payments, clock usage and transfers, and media content popularity.

In step 9-734 of Figure 9-7C, upon receiving the redirection command, the present embodiment causes the media playback application 9-501 (Figures 9-5A, 9-5B, 9-5C, and 9-5D) operating on client computer system 9-210 to automatically transmit to content server 9-251 a new media delivery request which can include the time sensitive access key and the address of the desired media content.

In step 9-736 of Figure 9-7C, content server 9-251 determines whether the time sensitive access key associated with the new media delivery request is valid. If content server 9-251 determines that the time sensitive access key is valid, the present embodiment proceeds to step 9-738 of Figure 9-7C. However, if content

server 9-251 determines that the time access key is not valid, the present embodiment proceeds to step 9-737, a client redirect.

In step 9-737, content server redirects client computer 9-210 to step 9-732 (not shown) where a new access key is generated. Alternatively, step 9-737 causes the present embodiment to return to step 9-704 of Figure 9-7A. In yet another embodiment, step 9-737 causes client computer system 9-210 to be disconnected from content server 9-251.

In step 9-738 of Figure 9-7C, content server 9-251 transmits the requested high fidelity media content to client computer system 9-210. It is noted that each media content file delivered to client computer system 9-210 can have a header attached thereto, prior to delivery, as described with reference to Figure 9-4. It is further noted that both the media content and the header attached thereto can be encrypted. In one embodiment, the media content and the header can be encrypted differently. Alternatively, each media content file encrypted differently. In another embodiment, groups of media files are analogously encrypted. It is noted that public domain encryption mechanisms, e.g., Blowfish, and/or non-public domain encryption mechanisms can be utilized.

Still referring to step 9-738, content server 9-251 transmits the requested media content in a burst load (in comparison to a fixed data rate), thereby transferring the content to client computer system 9-210 as fast as the network transfer rate allows. Further, content server 9-251 can have its download rate adapted to be equal to the transfer rate of the network to which it is coupled. In

another embodiment, the content server 9-251 download rate can be adapted to equal the network transfer rate of the client computer system 9-210 to which the media content is being delivered. For example, if client computer system 9-210 is coupled to Internet 9-201 via a T1 connection, then content server 9-251 transfers the media content at transmission speeds allowed by the T1 connection line. As such, once the requested media content is transmitted to client computer system 9-210, content server 9-251 is then able to transmit requested media content to another client computer system, e.g., 9-220 or 9-230. Advantageously, this provides an efficient means to transmit media content, in terms of statistical distribution over time and does not overload the communication network(s).

It is noted that delivery of the requested media content by content server 9-250 to client computer system 9-210 can be implemented in a variety of ways. For example, an HTTP (hypertext transfer protocol) file transfer protocol can be utilized to transfer the requested media content as well as a copyright compliance mechanism 9-300 to client 9-210. In this manner, the copyright compliance mechanism as well as each media content file/title can be delivered in its entirety. In another embodiment, content server 9-251 can transmit to client computer system 9-250 a large buffer of media content, e.g., audio clips, video clips, and the like.

In step 9-740 of Figure 9-7C, upon receiving the requested high fidelity media content from content server 9-251, the present embodiment causes client computer system 9-210 to store the delivered media content in a manner that is ready for presentation, e.g., play. The media content is stored in client computer system 9-210 in a manner that restricts unauthorized redistribution. For example, the present

embodiment can cause the high fidelity media content to be stored in a volatile memory device, utilizing one or more hidden directories and/or custom file systems that may be hidden, where it may be cached for a limited period of time.

Alternatively, the present embodiment can cause the high fidelity media content to be stored in a non-volatile memory device, e.g., 9-103 or data storage device 9-108. It is noted that the manner in which each of the delivered media content file(s) is stored, volatile or non-volatile, can be dependent upon the licensing restrictions and copyright agreements applicable to each media content file. It is further noted that in one embodiment, when a user of client computer system 9-210 turns the computer off or causes client computer system 9-210 to disconnect from the network, the media content stored in a volatile memory device is typically deleted therefrom.

Still referring to step 9-740, in another embodiment, the present embodiment can cause client computer system 9-210 to store the received media content in a non-volatile manner within a media application operating therein, or within one of its Internet browser applications (e.g., Netscape Communicator™, Microsoft Internet Explorer™, Opera™, Mozilla™, and the like) so that delivered media content can be used in a repetitive manner. Further, the received media content can be stored in a manner making it difficult for a user to redistribute in an unauthorized manner, while allowing the user utilization of the received media content, e.g., by utilizing one or more hidden directories and/or custom file systems that may also be hidden. It is noted that by storing media content with client computer system 9-210 (when allowed by applicable licensing agreements and copyright restrictions), content server 9-251 does not need to redeliver the same media content to client computer

system 9-210 each time its user desires to experience (e.g., listen to, watch, view, etc.) the media content file.

In step 9-742 of Figure 9-7C, the received media content file is then fed into a media player application (e.g., playback application 9-501 of Figures 9-5A, 9-5B, 9-5C, and 9-5D), which then runs it through a codec, e.g., coder/decoder 9-303 of CCM 9-300, in one embodiment. In response, coder/decoder 9-303 sends an authorization request to the server, e.g., 9-251, with attached authorization data, as described herein. In response to receiving codec's 9-303 authorization request, server 9-251 compares the received authorization data with that stored in server 9-251, and subsequently, the present embodiment proceeds to step 9-744.

In step 9-744, the server 9-251 responds with a pass or fail authorization. If server 9-251 responds with a fail, such that the received authorization data is invalid, the present method can proceed to step 9-745, where server 9-251 can, in one embodiment, notify the user of client system 9-210, e.g., by utilization of skin 9-306, that there was an unsuccessful authorization of the requested media content file. It is noted that alternative messages having similar meanings may also be presented to the user of client computer system 9-210, thereby informing the user that the delivery failed. However, if the authorization data passes, the present method proceeds to step 9-746.

In step 9-746, server 9-251 transmits certain data back to the media player application which enables the media player application to present the contents of the media file via media playback application 9-501 of Figures 9-5A, 9-5B, 9-5C, and 9-

5D. In one embodiment, a decryption key can be included in the transmitted data to decrypt the delivered media content file. In another embodiment, an encryption/decryption key can be included in the transmitted data to allow access to the contents of the media file. The present method then proceeds to step 9-748.

In step 9-748 of Figure 9-7C, subsequent to media file decryption, the media file may be passed through CCM 9-300, e.g., a coder/decoder 9-303, to a media player application operating on client computer system 9-210, e.g., playback application 9-501 of Figures 9-5A, 9-5B, 9-5C, and 9-5D, which can then access and utilize the delivered high fidelity media content, enabling its user(s) to experience the media content, e.g., listen to it, watch it, view it, or the like. In one embodiment of the present invention, a specialized or custom media player may be involved in order to experience the media content, e.g., skin 9-306 of Figure 9-3. Skin 9-306 may be implemented when CCM 9-300 cannot modify an industry standard media player application to comply with copyright restrictions and/or licensing agreements in accordance with the DMCA. Alternatively, a specialized or custom media player may not be needed to experience the media content. Instead, an industry standard media player can be utilized by client computer system 9-210 to experience the media content. Typically, many media player applications are available and can include, but are not limited to, Windows™ Media Player™ for PCs (personal computers), iTunes™ Player or QuickTime™ for Apple computers, and XMMS player for computers utilizing a Linux operating system. Regardless of the media player application utilized, while the media file is passed to the media player application, e.g., in a frame by frame basis or in a buffer by buffer basis,

coder/decoder 9-303 will repeatedly ensure that CCM 9-300 rules are being enforced at any particular moment during media playback, shown as step 9-750.

In step 9-750, as the media file content is delivered to the media player application, e.g., media player application 9-501 of Figures 9-5A, 9-5B, 9-5C, and 9-5D, periodically, e.g., after a specified number of frames, after a defined period of time, or any desired time or data period, coder/decoder 9-303 repeatedly determines whether or not all the rules are enforced, in accordance with rules as defined by CCM 9-300. If the rules are not enforced, e.g., change due to a user opening up a recording application (e.g., Total Recorder or alternative application) the present method proceeds to step 9-751. If the rules, in accordance with CCM 9-300, are enforced, the present method then proceeds to step 9-752.

In step 9-751 of Figure 9-7C, if the rules according to CCM 9-300 are not enforced, the presentation of the media content is, in one embodiment, suspended or halted. In one embodiment, CCM 9-300 can selectively control switches 9-311 and 9-511 (Figure 9-5A) to prevent output of incoming media 9-499 (Figures 9-5A, 9-5B, 9-5C, and 9-5D) to a recording application 9-502 (Figures 9-5A, 9-5B, and 9-5C, via wave shim driver 9-309 and direct sound 9-504 respectively, thus preventing unauthorized recording of incoming media 9-499. In another embodiment, CCM 9-300 can selectively control switches 9-311 and 9-312 (Figure 9-5B) to prevent output of incoming media 9-499 to recording application 9-502 via wave shim driver 9-309 and custom media device 9-310, thus preventing unauthorized recording of incoming media 9-499. In yet another embodiment, CCM 9-300 can selectively control switches 9-311, 9-312, to not only prevent incoming media 9-499 from being

recorded in an unauthorized manner but can also selectively control switch 9-571 (Figure 9-5C) to prevent unauthorized output of incoming media 9-499 via digital output 9-575 of media hardware output device 9-570. In yet another embodiment, CCM 9-300 can selectively control switches 9-311, 9-312, 9-571, and 9-511 to a prevent kernel streaming mechanism 9-515, e.g., DirectKS of Figure 9-5D, which can establish a connection with media device driver 9-505 of Figure 9-5D, from capturing incoming media content and returning it to a recording application (e.g., 9-502) to create an unauthorized recording of the media content. In one embodiment, incoming media 9-499 may not be output from digital output 9-575. In another embodiment, incoming media 9-499 may be output via digital output 9-575 but in an inaudible manner, e.g., silence. In yet another embodiment, incoming media 9-499 be audible but recording functionality can be disabled, such that the media content cannot be recorded.

In step 9-752, if the rules are enforced in accordance with CCM 9-300, coder/decoder 9-303 retrieves a subsequent portion of the media content that is stored locally in client computer system 9-210. The newly retrieved portion of the media file is then presented by the client's media player application, shown in the present method as step 9-748. While the newly retrieved portion is presented, embodiments of the present method then again perform step 9-750, then step 9-752 or 9-751, then step 9-748, then 9-750, etc., in a continual loop until the media file contents are presented in their entirety. Advantageously, by constantly monitoring playing media files, CCM 9-300 can detect undesired activities and enforce those rules defined by CCM 9-300.

Figure 9-8 is a diagram of an exemplary high-speed global media content delivery system 9-800, in accordance with one embodiment of the present invention. In one embodiment, system 9-800 can be utilized to globally deliver media content, e.g., audio media, video media, graphic media, multimedia, alphanumeric media, etc., to a client computer system, e.g., 9-210, 9-220, and/or 9-230, in conjunction with a manner of delivery similar to that described herein. In one embodiment, system 9-800 includes a global delivery network 9-802 that can include multiple content servers, e.g., 9-804, 9-806, 9-808, 9-810, 9-812, 9-814, and 9-816, that can be located throughout the world and which may be referred to as points of presence or media delivery point(s). Each of content server 9-804 to 9-816 can store a portion, a substantial portion, or the entire contents of a media content library that can be delivered to client computer systems via a network, e.g., Internet 9-201, or a WAN (wide area network). Accordingly, each of content server 9-804 to 9-816 can provide media content to of client computer systems in its respective vicinity in the world. Alternatively, each content server can provide media content to a substantial number of client computer systems

For example, a media delivery point (MDP) 9-816, located in Tokyo, Japan, is able to provide and deliver media content from the media content library stored in its content database, e.g., 9-451, to client computer systems within the Asiatic regions of the world while a media delivery point 9-812, located in New York City, New York, USA, is able to provide and deliver media content from its stored media content library to client devices within the Eastern United States and Canada. It is noted that each city name, e.g., London, Tokyo, Hamburg, San Jose, Amsterdam, or New York, associated with one of the media delivery points 9-804 to 9-816 represents the

location of that particular media delivery point or point of presence. However, it is further noted that these city names are exemplary because media delivery points 9-804 to 9-816 can be located anywhere within the world, and as such are not limited to the cities shown in global network 9-802.

Still referring to Figure 9-8, it is further noted that global system 9-802 is described in conjunction with Figures 9-2, 9-3, 9-4, 9-5A to 9-5D, and 9-6, in order to more fully describe the operation of embodiment of the present invention. Particularly, subsequent to a client computer system, e.g., client computer system 9-210 of Figure 9-2, interacting with a web server, e.g., web server 9-250 of Figure 9-2, as described herein, web server 9-250, in one embodiment, can redirect client computer system 9-210 to receive the desired media content from an MDP (e.g., 9-804 to 9-816) based on one or more differing criteria.

For example, computer system 9-210 may be located in Brattleboro, Vermont, and its user causes it to log-in with a web server 9-250 which can be located anywhere in the world. It is noted that steps 9-702 to 9-730 of Figures 9-7A and 9-7B can then be performed as described herein such that the present embodiment proceeds to step 9-732 of Figure 9-7C. At step 9-732, the present embodiment can determine which media delivery points, e.g., 9-804, 9-806, 9-808, 9-810, 9-812, 9-814, or 9-816, can subsequently provide and deliver the desired media content to client computer system 9-210.

Still referring to Figure 9-8, one or more differing criteria can be utilized to determine which media delivery point to select for delivery of the desired media

content. For example, the present embodiment can base its determination upon which media delivery point is in nearest proximity to client computer system 9-210, e.g., media delivery point 9-816. This can be performed by utilizing the stored registration information, e.g., address, provided by the user of client computer system 9-210. Alternatively, the present embodiment can base its determination upon which media delivery point provides media content to the part of the world in which client computer system is located. However, if each media delivery point (e.g., 9-804 to 9-816) stores differing media content, the present embodiment can determine which one can actually provide the desired media content. It is noted that these are exemplary determination criteria and the embodiments of the present invention are not limited to such implementation.

Subsequent to determination of which media delivery point is to provide the media content to client computer system 9-210 at step 9-732, web server 9-250 transmits to client computer system 9-210 a redirection command to media delivery point/content server 9-812 along with a time sensitive access key, also referred to as a session key, (e.g., for that hour, day, or any defined time frame) thereby enabling client computer system 9-210 to eventually receive the requested media content. Within system 9-800, the redirection command can include a time sensitive address of the media content location within media delivery point 9-812. Accordingly, the New York City media delivery point 9-812 can subsequently provide and deliver the desired media content to client computer system 9-210. It is noted that steps 9-732 to 9-742 and step 9-737 of Figure 9-7C can be performed by media delivery point 9-812 in a manner similar to content server 9-251 described herein.

Advantageously, by utilizing multiple content servers, e.g., media delivery point 9-804 to 9-816, to provide high fidelity media content to client computer systems, e.g., 9-210 to 9-230, located throughout the world, communication network systems of the Internet 9-201 do not become overly congested. Additionally, global network 9-802 can deliver media content to a larger number of client computer systems (e.g., 9-210 to 9-230) in a more efficient manner. Furthermore, by utilizing communication technology having data transfer rates of up to 9-320 Kbps (kilobits per second) or higher, embodiments of the present invention provide for rapid delivery of the media content in a worldwide implementation.

Referring still to Figure 9-8, it is noted that media delivery points/content servers 9-804 to 9-816 of global network 9-802 can be coupled in a wide variety of ways in accordance with the present embodiment. For example, media delivery point 9-804 to 9-816 can be coupled utilizing wired and/or wireless communication technologies. Further, it is noted that media delivery points 9-804 to 9-816 can be functionally coupled such that if one of them fails, another media delivery point can take over and fulfill its functionality. Additionally, one or more web servers similar to web server 9-250 can be coupled to global network 9-802 utilizing wired and/or wireless communication technologies.

Within system 9-800, content server/media delivery point 9-804 includes a web infrastructure that, in one embodiment, is a fully redundant system architecture. It is noted that each MDP/content server 9-806 to 9-816 of global network 9-802 can be implemented to include a web infrastructure in a manner similar to the implementation shown in MDP 9-804.

Specifically, the web infrastructure of media delivery point 9-804 includes firewalls 9-818 and 9-820 which are each coupled to global network 9-802. Firewalls 9-818 and 9-820 can be coupled to global network 9-802 in diverse ways, e.g., utilizing wired and/or wireless communication technologies. Particularly, firewalls 9-818 and 9-820 can each be coupled to global network 9-702 via a 10/100 Ethernet handoff. However, system 9-800 is not limited in any fashion to this specific implementation. It is noted that firewalls 9-818 and 9-820 are implemented to prevent malicious users from accessing any part of the web infrastructure of media delivery point 9-804, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 9-840 coupled to device 9-836 while firewall 9-820 includes a device 9-838, e.g., a router or other switching mechanism, coupled therewith and a DB (database) server 9-842 coupled to device 9-838. Furthermore, DB server 9-840 is coupled with device 9-838 and DB server 9-842 is coupled with device 9-836.

Still referring to Figure 9-8, and within media delivery point 9-804, firewall 9-818 is coupled to a director device 9-822 which is coupled to internal web application server 9-826 and 9-828, and a hub server 9-830. Firewall 9-820 is coupled to a director 9-824 which is coupled to internal web application servers 9-826 and 9-828, and hub server 9-830. Hub server 9-830 can be implemented in a variety of ways including, but not limited to, as a Linux hub server. Hub server 9-780 is coupled to a data storage device 9-832 capable of storing media content. Data storage device 9-832 can be implemented in a variety of ways, e.g., as a RAID (redundant array of inexpensive/independent disks) appliance.

It is noted that media delivery points 9-804 to 9-816 can be implemented in any manner similar to content server 9-250 described herein. Additionally, media delivery points 9-804 to 9-816 of the present embodiment can each be implemented as one or more physical computing devices, e.g., computer system 9-100 of Figure 9-1.

In another embodiment, CCM 9-300 can be adapted to be disposed on a media storage device, e.g., media storage device 9-999 of Figures 9-10 and 9-11. Media storage device 9-999 can be, but is not limited to, a CD, a DVD, or other optical or magnetic storage device. By virtue of disposing a version of CCM 9-300 on a media storage device 9-999, embodiments of the present invention can provide copy protection for audio, video, multimedia, graphics, information, data, software programs, and other forms of media that may contain copyrighted material and which may be disposed on a media storage device. Alternatively, CCM 9-300 can be adapted to be installed on a computer system, e.g., client computer system 9-210, via a media storage device 9-999 upon which it may be disposed.

Figure 9-9 is a block diagram of a copyright compliance mechanism/media storage device (CCM/MSD) 9-900, a version of CCM 9-300 adapted to be disposed on a media storage device, e.g., media storage device 9-999 of Figures 9-10 and 9-11. It is noted that CCM 9-300 in CCM/MSD 9-900 is analogous to CCM 9-300 as described in Figures 9-3, 9-4, 9-5A-D, 9-6A and 9-7A to 9-7C. Further, CCM/MSD 9-900 can be readily updated in accordance with global delivery system 9-800, as described in Figures 9-7A to 9-7C, and Figure 9-8.

In one embodiment, CCM/MSD 9-900 is adapted to provide stand-alone compliance with copyright restrictions and licensing agreements applicable to media files that may be disposed on a media storage device, e.g., media storage device 9-999. In another embodiment CCM/MSD 9-900 is adapted to be installed on a computer system, e.g., client computer system 9-210 to provide compliance with copyright restrictions and licensing agreements applicable to media files as described in Figures 9-3, 9-4, 9-5A to 9-5D, 9-6A and 9-7A to 9-7C.

Referring to Figure 9-9, CCM/MSD 9-900 includes an autorun protocol component 9-910 for invoking automatic installation of CCM 9-300. To deter users from attempts at defeating various features inherent to CCM 9-300, e.g., the autorun feature, CCM 9-300's monitoring program, agent program 9-304, verifies that those features that are to be operational are operational, and if not, CCM 9-300 prohibits the user from experiencing the contents of the media storage device.

If a user somehow defeats the autorun feature, and the user attempts to utilize an application to capture an image of the content, the application will make an image of the content on the media storage device, which also images the copyright protection contained thereon, and when the image is played, CCM 9-300 recognizes the copy protection is present, and CCM 9-300 will only allow the user to experience the content when authorized, once CCM 9-300 is installed.

By virtue of the protections as described above provided by CCM 9-300, users will be able to experience the content of the media storage device in the

content's original high quality format, thereby obviating the need to compress the media file used on client system 9-210.

Advantageously, the user will no longer need to suffer through poor quality output as a result of severely compressed media files.

It is noted that when adapted to be implemented in conjunction with a secure file format, meaning that the format of the file is, without proper authorization, non-morphogenic, embodiments of the present invention also provide effective compliance with copyright restrictions and licensing agreements with secure files formats. CCM 9-300 can control the types of file formats into which the media file can be transformed, e.g., .wav, .mp3, etc.

In one embodiment, the autorun feature associated with media storage device drive 9-1112 of client system 9-210 is activated and operational. Alternatively, a notice of required autorun activation within client system 9-210 may be displayed on the media storage device and/or the case in which the media storage device is stored.

In another embodiment, if CCM 9-300 is present or if the user is coupled to a server, then messages containing instructions on how to activate the autorun feature of client system 9-210 may be presented to the user.

In one embodiment autorun protocol component 9-910 can detect media storage device drives resident on a computer system, e.g., client computer system 9-210.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 9-910 for detecting media storage device drives residing and operable on client computer system 9-210, according to one embodiment of the present invention.

```

if ( (dwRetVal = GetLogicalDrives())
    != (DWORD) 0)
{
    /* initialize variables */
    dwMask = (DWORD) 1;

    /* initialize path to root of current drive */
    _tcscopy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <<= 1)
    {
        if ((dwRetVal & dwMask) != 0)
        {
            /* construct path to root of drive */
            szDrive[0] = (TCHAR) 'A' + nIndex;

            if (GetDriveType(szDrive) == DRIVE_CDROM)
            {
                MessageBox((HWND) 0,
                    _T("CD-ROM drive found."),
                    szDrive,
                    MB_OK);
            }
            else
            {
                /* clear bit at current position */
                dwRetVal &= (~dwMask);
            }
        }
    }
}

```

In another embodiment, autorun protocol component 9-910 can detect whether a media storage device containing media files has been inserted into a media storage device drive coupled with client computer system 9-210, e.g., drive 9-1112 of Figure 9-10. In another embodiment, CCM 9-300 can include instructions for monitoring media storage device drive 9-1112, and upon detection of drive activation, CCM 9-300 determines what type of media storage device has been inserted therein. Subsequently, CCM 9-300 can detect various triggers on the media storage device to invoke its protection, e.g., a hidden file on newer media storage devices and/or the copyright indicator bit on legacy media storage devices, obviating the need for autorun. Upon detection, CCM 9-300 can invoke the appropriate protection for the associated media file.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 9-910 for detecting a media storage device inserted in a media storage device drive residing and operable on client computer system 9-210, according to one embodiment of the present invention.

```

    /* set error mode for operation */
    uiErrMode = SetErrorMode(SEM_FAILCRITICALERRORS);

    /* initialize path to root of current drive */
    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <=< 1)
    {
        if ((dwCDROMMask & dwMask) != 0)
        {
            /* construct path to root of drive */
            szDrive[0] = (TCHAR) 'A' + nIndex;

```

```

        if ( GetDiskFreeSpace(szDrive,
                                &dwSectors,
                                &dwBytes,
                                &dwClustersFree,
                                &dwClusters)
            != 0)
        {
            /* add bit for drive to mask */
            dwRetVal |= dwMask;
        }
    }

    /* restore original error mode */
    SetErrorMode(uiErrMode);

```

Additionally, autorun protocol component 9-910 can also detect changes in media, e.g., insertion of a different media storage device 9-999. Further, other media changes can be detected subsequent to adaptation of the source code including, but not limited to, detecting a previously accessed media file, detecting a previously inserted media storage device.

The following C++ source code is an exemplary implementation of a portion of autorun protocol component 9-910 for detecting a change in media, according to one embodiment of the present invention.

```

    /* initialize path to root of current drive */
    _tcscpy(szDrive, _T("A:\\"));

    for (nIndex = 0, dwMask = (DWORD) 1;
        dwMask != (DWORD) 0;
        nIndex++, dwMask <= 1)
    {
        /* check for presence of CD-ROM media in drive */
        if ((dwCurrMask & dwMask) != 0)
        {
            /* check if media previously in drive */
            if ((dwPrevMask & dwMask) == 0)

```

```

    {
        /* construct path to root of drive */
        szDrive[0] = (TCHAR) 'A' + nIndex;

        /* check for presence of marker on drive */
        if (IsMPBMarkerPresent(szDrive) != 0)
        {
            /* process autorun information present on drive */
            nRetVal = ProcessAutorun(szDrive);
        }
    }
}

```

Still referring to Figure 9-9, CCM/MSD 9-900 also includes a kernel level filter driver 9-920 for controlling a data input path of an operating system coupled with and operable on client computer system 9-210.

CCM/MSD 9-900 also includes a generalized filter driver 9-930 for controlling ripping and "burning" applications, e.g., Nero, Roxio, Exact Audio Copy, and others, thereby preventing such activities.

The following C++ source code is an exemplary implementation of a portion of generalized filter driver 9-930 for controlling ripping and burning applications that may be residing on and operable within client computer system 9-210, in accordance with one embodiment of the present invention.

```

bool    bDisabled;          /* flag indicating CD reads disabled */

/* initialize variables */
bDisabled = false;

```

```

    if (bProtected == true)
    {
        if (type == IRP_MJ_DEVICE_CONTROL)
        {
            ULONG ulloControlCode = stack-
>Parameters.DeviceIoControl.IoControlCode;

            if (ulloControlCode == IOCTL SCSI_PASS_THROUGH)
            {
                SCSI_PASS_THROUGH * pspt =
(SCSI_PASS_THROUGH *)
Irp->AssociatedIrp.SystemBuffer;

                if ( (pspt != NULL)
                    && (pspt->Cdb[0] == SCSIOP_READ_CD))
                {
                    pspt->DataTransferLength = 0;
                    pspt->ScsiStatus = 0;

                    bDisabled = true;
                }
            }
            else if (ulloControlCode ==
IOCTL SCSI_PASS_THROUGH_DIRECT)
            {
                SCSI_PASS_THROUGH_DIRECT * psptd =
(SCSI_PASS_THROUGH_DIRECT *)
Irp->AssociatedIrp.SystemBuffer;

                if ( (psptd != NULL)
                    && (psptd->Cdb[0] == SCSIOP_READ_CD))
                {
                    psptd->DataTransferLength = 0;
                    psptd->ScsiStatus = 0;

                    bDisabled = true;
                }
            }
        }
    }

    if (bDisabled == true)
    {
        /* complete current request */
        status = CompleteRequest(Irp, STATUS_SUCCESS, 0);
    }
    else
    {
        /* pass request down without additional processing */

```

```
status = IoAcquireRemoveLock(&pdx->RemoveLock, Irp);

if (!INT_SUCCESS(status))
    return CompleteRequest(Irp, status, 0);

IoSkipCurrentIrpStackLocation(Irp);
status = IoCallDriver(pdx->LowerDeviceObject, Irp);
IoReleaseRemoveLock(&pdx->RemoveLock, Irp);
}
```

Still referring to Figure 9-9, CCM/MSD 9-900 includes a CCM 9-300, analogous to CCM 9-300 of Figure 9-3, that is adapted to be installed in client computer system 9-210 in the manner described herein.

In one embodiment, kernel level filter driver 9-920, generalized filter driver 9-930 and CCM 9-300 of CCM/MSD 9-900 are automatically installed on client computer system 9-210, subsequent to insertion of media storage device 9-999 into a media storage device drive, e.g., media storage device drive 9-1112 of Figures 9-10 and 9-11. Autorun protocol component 9-910, as described above, detects insertion of media storage device 9-999 into an appropriate drive, and initiates installation of the components, e.g., CCM 9-300, driver 9-920 and driver 9-930. In one embodiment, drivers 9-920 and 9-930 may be temporarily installed and may be deleted upon removal of media storage device 9-999 from media storage device drive 9-1112. In yet another embodiment, drivers 9-920 and 9-930 may be installed in hidden directories and/or files within client computer system 9-210. In another embodiment, some components of CCM 9-300 can remain installed on client system 9-210, e.g. the monitoring program (agent program 9-304). In still another embodiment, other components, e.g., the kernel level filter driver 9-920, can be

dynamically loaded and unloaded as necessary in accordance with copyright restrictions and licensing agreements applicable to the media file.

Embodiments of the present invention utilize software, e.g., CCM/MSD 9-900, that is placed on media storage device 9-999, in conjunction with controlling software CCM 9-300 installed on client computer system 9-210, and web server 9-250 and/or content server 9-251, wherein each component is communicatively coupled with the other via the Internet, thereby enabling dynamic updating of CCM 9-300 in the manner as described with reference to Figure 9-4, and steps 9-716 and 9-718 of Figures 9-7A to 9-7C.

In the present embodiment, CCM/MSD 9-900 provides a stand alone DRM that is far more sophisticated than existing DRM solutions. This is because CCM/MSD 9-900 goes into the data pathway of the operating system operable on client computer system 9-210 and obtains control of the data pathway, e.g., filter driver 9-1108 of Figure 9-11, rather than exploiting inefficiencies or errors in the computer system.

Figure 9-10 is a block diagram of a communicative environment 9-1000 for controlling unauthorized reproduction of protected media files disposed on a media storage device. Included in communicative environment 9-1000 is a media storage device drive 9-1112 coupled with a client computer system 9-210 via a data/address bus 9-110. Client computer system 9-210 is coupled with web server 9-250 and content server 9-251 via Internet 9-201. A media storage device 9-999, upon which a CCM/MSD 9-900 may be disposed, is inserted in media storage device drive 9-

1112. Autorun protocol component 9-910 detects the insertion and automatically invokes installation of CCM 9-300, kernel level filter driver 9-920 and generalized filter driver 9-930 from media storage device 9-999 into client computer system 9-210. Subsequent to installation, CCM 9-300 initiates a dynamic update with web server 9-250 and/or content server 9-251, via Internet 9-201. By installing CCM 9-300 on client computer system, agent program 9-304 (Figure 9-3) of CCM 9-300 is able to control the integrity of the software. Additionally, by conferring with servers 9-250 and/or 9-251 via Internet 9-201 online, the CCM 9-300 software version on media storage device 9-999 and installed on client computer system 9-210 can be updated when circumventions occur and kept current from platform to platform.

Advantageously, the monitoring mechanism of agent program 9-304 enables constant morphing of the version of CCM 9-300 disposed on media storage device 9-999 by communicating with server 9-250 and/or 9-260 and utilizing the dynamic update capabilities of global network 9-800 to readily update that which has been installed on client computer system 9-210, via media storage device 9-999.

In one embodiment, the installation is performed clandestine with respect to the user and is initiated by inserting media storage device 9-999 into an appropriate media storage device drive, e.g. a magnetic/optical disk drive or alternative device drive coupled with client system 9-210. If the user is not registered with CCM 9-300, as described herein with reference to Figure 9-4 and Figures 9-7A to 9-7C, once installed, CCM 9-300 initiates an update process with web server 9-250 and/or content server 9-251 to readily include updates that have been invoked subsequent to release of the media file on media storage device 9-999. By virtue of the dynamic

update capabilities of CCM 9-300, regardless of the version of CCM 9-300 on media storage device 9-999, CCM 9-300 provides compliance with copyright restrictions and licensing agreements applicable to the media file on media storage device 9-999. Advantageously, enabling dynamic adaptability of CCM 9-300 provides for continued interoperability with new and updated operating systems, advancements in electronic technology, communication technologies and protocols, and the like, ensuring the effectiveness of CCM 9-300 into the future.

In another embodiment, if the user is a registered user with global delivery system 9-800, CCM 9-300 can detect which version is most current. Accordingly, when the version existing on client system 9-210 is more current than the version (for install) on media storage device 9-999, CCM 9-300 can bypass the install process and present the contents contained on media storage device 9-999 to the user for them to experience.

Further advantageously, this technology is backward compatible with media storage device drives manufactured subsequent to 1982. Additionally, CCM 9-300 is compatible with media storage devices having a copyright indicator bit disposed thereon. The copyright indicator bit has been included on all CDs released since 1982.

In the present embodiment of Figure 9-10, the media file is not encrypted on media storage device 9-999. In one embodiment, if the media file is encrypted on computer 9-210, it can be decrypted on the computer 9-210. However, home players and/or stand alone media playing devices rarely include a decryption

mechanism, and to experience the music on a home machine, the music is conventionally not encrypted.

In one embodiment, an additional component of CCM 9-300 is that the trigger for agent program 9-304 may be the copyright bit indicator. This means when the copyright indicator bit is detected by CCM 9-300, the functions of CCM 9-300 are initiated. Alternatively, in another embodiment, when the copyright bit indicator is not detected, CCM 9-300 may remain in an un-invoked or idle state. If CCM 9-300 can detect the copyright bit indicator, CCM 9-300 can provide the appropriate compliance with regard to copyright restrictions and licensing agreements applicable to the media files.

In an alternative embodiment, a trigger control in the table of contents of a media storage device 9-999 includes instructions for triggering autorun protocol 9-910 of CCM/MSD 9-900 and can utilize the copyright indicator bit or alternative implementation to trigger the technology. In this manner, CCM 9-300 can control copyrighted works while public domain material can be experienced and reproduced at a user's discretion. Because autorun is problematic for media storage device manufacturers, embodiments of CCM/MSD 9-900 can include alternative autorun programs that perform analogous to autorun.

In another embodiment, CCM 9-300 can invoke its own proprietary player, e.g., custom media device 9-310 as described with reference to Figure 9-3, thus enabling increased control of copyright restrictions and/or licensing agreements applicable to the media. By invoking custom media device 9-310, CCM 9-300

enables user experience of the media while providing protection against unauthorized reproduction of the media disposed on media storage device 9-999. .

In an alternative embodiment, the media files and the CCM/MSD 9-900 disposed on a media storage device 9-999 are encrypted. This implementation is particularly advantageous for demonstration (demo) versions of media files, beta test versions, and the like that may be disposed on media storage device 9-999. It is noted that the present embodiment is operable in an online environment, meaning that client computer system 9-210 is communicatively coupled with web server 9-250 and/or content server 9-251 to enable a user experience of the content on a demo version of media storage device 9-999. In this implementation, CCM 9-300 allows for specific plays for specific users, which can be controlled via a network, e.g., network 9-1000 of Figure 9-10, and server 9-250 and/or 9-251.

In another embodiment, CCM 9-300 can be implemented for demo and/or pre-release protection. In this embodiment, CCM 9-300 utilizes sophisticated encryption technology to encrypt the table of contents and CCM 9-300 with an associated decrypted key located on client computer system. Encrypting CCM 9-300 can also deter nefarious attempts to reverse engineer CCM 9-300. Decryption can be performed using an associated decryption key. Alternatively, decryption can be performed by a proprietary or custom media player application resident on demo media storage device, e.g., 9-999.

The content of media storage device 9-999 is encrypted, using various levels of encryption to provide protection levels commensurate with copyright holders

desires and required protection. For example, media storage device 9-999 is delivered to a user or critic for the purposes of review, the user inserts media storage device 9-999 into the appropriate storage device reader or connector coupled with the journalist's computer, and CCM 9-300 is installed on client system 9-200 in a manner clandestine to the user. Once installed, CCM 9-300 initiates a communication session with web server 9-250/content server 9-251, where content server 9-251 can provide authorization for the user to experience the media on media storage device 9-999.

Accordingly, if the user, to whom demo media storage device 9-999 had been released, had demo media storage device 9-999 stolen, or if the user allowed alternative parties try to experience the content of media storage device 9-999, the unauthorized party would have to try to crack the encryption keys and the encryption of the actual content of media storage device 9-999, consuming non-trivial amounts of time.

Thus, CCM 9-300 is able to control which users receive authorization to experience the media of media storage device 9-999, how many times the user may experience the media, and CCM 9-300 may also define a period of time until the media may no longer be accessible. This may enable copyright holders to release the content on an authorized media storage device, e.g., 9-999, prior to pirated copies flooding the market.

Accordingly, a demo media storage device 9-999 may be configured such that a first user may get a copy, a second user may get a copy, and if it is known that

the second user will share the demo with a third and a fourth user, then the known users would be enabled to experience the media. Advantageously, by virtue of defining which users can access and experience the media, any unauthorized sharing of the media by one of the authorized users can be readily detected, and further sharing or experiencing of the media may be halted. Additionally, because the authorized user shared the media in an unauthorized manner, in a worse case scenario, criminal charges could be filed against that user.

It is noted that placing CCM/MSD 9-900 on a media storage device, e.g., 9-999, so as to enable installation of CCM 9-300 on client system 9-210 is one manner in which CCM 9-300 can be installed on client system 9-210. An alternative manner in which CCM 9-300 can be installed on client computer system 9-210 is through "cross-pollination." For example, webcasters broadcast the media file to the user. The media file has a CCM 9-300 coupled with the media file, and upon downloading the media file onto client computer system 9-210, embodiments of the present invention enable the installation of CCM 9-300 onto client computer system 9-210. In another manner, CCM 9-300 is incorporated into and becomes part of an operating system operational on client system 9-210. Alternatively, laws are passed that mandate the inclusion of CCM 9-300 on each client computer system 9-210.

Figure 9-11 is an exemplary logic/bit path block diagram 9-1100 of a client computer system, e.g., 9-210, configured with a copyright compliance mechanism (CCM) 9-300 for preventing unauthorized reproduction of copyrighted media according to an embodiment of the present invention. Copyright compliance mechanism 9-300 is, in one embodiment, coupled with and operational on client

system 9-210 in any manner described with reference to Figures 9-4, 9-5A to 9-5D, 9-6A, and 9-7A to 9-7C, 9-9, and 9-10.

Diagram 9-1100 of Figure 9-11 includes a media storage device media extraction/creation application 9-1102 communicatively coupled to operating system input/output subsystem 9-1104 via wave in line 9-1121 and wave out line 9-1138. Operating system input/output subsystem 9-1104 is coupled with media storage device class driver 9-1106 via wave in line 9-1123 and wave out line 9-1136. Media storage device class driver 9-1106 is coupled with filter driver 9-1108 via wave in line 9-1125 and wave out line 9-1134. Filter driver 9-1108 is coupled with media storage device port driver 9-1110 via wave in line 9-1127 and wave out line 9-1132. Filter driver 9-1108 is shown to include a switch 9-1111, controlled by CCM 9-300 via coupling 9-1160. Media storage device port driver 9-1110 is coupled with media storage device drive 9-1112 via wave line in 9-1129 and wave line out 9-1130. Media storage device 9-999, shown to include CCM/MSD 9-900 is receivable by media storage device drive 9-1112. Additionally, CCM 9-300 is coupled with operating system input/output subsystem 9-1104 via wave in line 9-1150 and wave out line 9-1151.

In one embodiment, CCM 9-300 is coupled to and controls selectable switch 9-1111 in filter driver 9-1108. Depending upon the copyright restrictions and/or licensing agreements applicable to a media file disposed on media storage device 9-999, CCM 9-300 controls whether switch 9-1111 is open (shown), thus preventing the media file from reaching media extraction/creation application 9-1102, or closed (not shown) so as to allow reproduction of the protected media file. Media

extraction/creation application 9-1102 can be a ripping or burning application such as Nero, Roxio, Exact Audio Copy, or other readily available application.

Continuing with Figure 9-11, media storage device 9-999 is received by media storage device drive 9-1112. CCM 9-300 determines whether media storage device 9-999 or media disposed thereon is protected by any copyright restrictions and/or licensing agreements, e.g., via detection of a copyright indicator bit. CCM 9-300 communicates with filter driver 9-1108 to control switch 9-1111 accordingly. In the present example, reproducing media storage device 9-999, and/or the contents thereon, would violate applicable restrictions and/or agreements and therefore switch 9-1111 is in an open position such that the output path to media extraction/creation application 9-1102, e.g., wave-out line 9-1138, is effectively blocked thereby preventing unauthorized reproduction of media storage device 9-999.

It is particularly noted that by virtue of CCM 9-300 controlling switch 9-1111, and therefore controlling wave-out line 9-1138, any incoming copyright protected media disposed on a media storage device 9-999 can be prevented from being reproduced in an unauthorized manner in accordance with applicable copyright restrictions and/or licensing agreements related to the incoming media.

Advantageously, as new secure or proprietary file formats are developed, CCM 9-300 can be readily adapted to be functional therewith. Further, CCM/MSD 9-900 can prevent users from making unauthorized reproductions of media files, recording, copying, ripping, burning, etc. By using kernel level filter drivers, e.g., filter driver 9-1108, and getting to a low enough level within the operating system

(OS) on client system 9-210, CCM 9-300 can detect particular applications and when they request media storage device drive 9-1112 to poll the media file for copying, ripping, etc., and disable the data input path. CCM 9-300, in this embodiment, deals with the input pathway.

In one embodiment, alternative applications that monitor the state of client computer system 9-210 can enable the autorun functionality of client computer system 9-210 or alternatively, invoke an automatic mechanism similar to autorun to ensure invocation of CCM 9-300 for compliance of copyright restrictions and/or licensing agreements applicable to media storage device 9-999 and/or the copyright protected media disposed thereon.

In one embodiment, CCM 9-300 can invoke a proprietary media player from media storage device 9-999, or activate a proprietary media player resident and operable on client computer system 9-210, or an alternative authorized media player resident on client computer system 9-210, as described herein with reference to Figure 9-3.

When media storage device 9-999 is a multisession device, e.g., a compact disk having a data session and a music session (audio tracks), and it is inserted into media storage device drive 9-1112, CCM 9-300 looks at the contents of the media storage device 9-999, and in some operating systems the audio tracks will not be displayed. Instead, the data session is shown, as is an autorun file, e.g., autorun protocol component 9-910, and upon clicking, invokes a player application. CCM 9-

300 can have a data session and files to which a user may not have access unless a player application is invoked.

In one embodiment, the player application could deposit a monitoring portion (e.g., agent program 9-304) on client system 9-210, which in one embodiment may reside on client computer system 9-210 subsequent to removal of media storage device 9-999 from media storage device drive 9-1112.

By virtue of content in a multisession media storage device 9-999, which may not be directly accessible to most player applications, at some point the player application will be invoked which can then install the CCM 9-300 into client system 9-210, according to one embodiment of the present invention.

In one embodiment, a proprietary media player application is stored on media storage device 9-999. However, it is not automatically invoked. Upon some user intervention, e.g., inserting media storage device 9-999 into media storage device drive 9-1112, the media player application is loaded onto client system 9-210 which has CCM 9-300 integrated therewith. Thus, CCM 9-300 is launched regardless of autorun being activated or not activated, and mandates the user to utilize the proprietary media player application to experience the content of the media files on the media storage device. 9-999.

In an alternative embodiment, client computer system 9-210 has autorun off, wherein it is common for the user to be unable to play a media file unless a proprietary media player application is invoked. Activating the proprietary media

player application can initiate an installation of those components of CCM 9-300 that are bypassed when autorun is not active.

Advantageously, by providing a copyright compliance mechanism, e.g., 9-300, which can be easily and readily installed on a client computer system, e.g., 9-210, embodiments of the present invention can be implemented to control access to, the delivery of, and the user's experience with media content subject to copyright restrictions and/or licensing agreements, for example, as defined by the DMCA. Additionally, by closely associating a client computer system, e.g., 9-210, with the user thereof and the media content they receive, embodiments of the present invention further provide for accurate royalty recording.

Figure 9-12 is a block diagram of a usage compliance mechanism 9-1200, an alternative version of copyright compliance mechanism 9-300 which is configured to be disposed on a media storage device, e.g., media storage device 9-999 of Figures 9-10, 9-11, 9-13, 9-14, and 9-15 in one embodiment of the present invention. It is noted that CCM 9-300 in usage compliance mechanism 9-1200 is analogous to CCM 9-300 as described herein with reference to Figures 9-3, 9-4, 9-5A to 9-5D, 9-6A, 9-7A to 9-7C, 9-8, 9-9, 9-10, and 9-11. Further, usage compliance mechanism 9-1200 can be readily updated in accordance with global delivery system 9-800, as described herein with reference to Figures 9-7A to 9-7C, and Figure 9-8.

In one embodiment, usage compliance mechanism 9-1200 is adapted to be disposed on a media storage device 9-999. Content disposed thereon can, in one embodiment, be demonstration and/or pre-release content. Examples of demonstration and/or pre-release content can include, but is not limited to, audio, video, multimedia, graphics, information, data, software programs, etc. Demonstration and/or pre-release content can contain, but is not limited to, digital movies or music that may be distributed to persons in the related media field for review, e.g., an motion picture academy member for their review of a movie, a record industry critic to review songs that may be released on a new compact disc, etc. Alternatively, demonstration and/or pre-release content can also contain, but is not limited to, a beta version of a software program, and the like.

Alternatively, the content disposed on media storage device 9-999 can, in another embodiment, be a commercial release of audio content, video content, software application, etc. Embodiments of the present invention are well suited to be implemented in a commercial environment, e.g., public presentation systems such as those in movie theaters, auditoriums, arenas and the like. Additionally, embodiments of the present invention are readily adaptable to be implemented in commercial distribution points, e.g., audio, video, and/or software retail and/or rental establishments, as well as for pay-per-view and/or pay-per-play implementations.

Further, literary works, documents, graphics such as pictures, painting, drawing, and the like can comprise the content on a media storage device. It is noted that a nearly endless variety of demonstration, pre-release, and/or commercially released content can be disposed on a media storage device 9-999.

Referring to Figure 9-12, usage compliance mechanism (UCM) 9-1200 includes an autorun protocol 9-910 for invoking installation of components of UCM 9-1200 on a client computer system, e.g., 9-210, in one embodiment of the present invention. Autorun protocol 9-910 of Figure 9-12 is analogous to autorun protocol 9-910 of Figure 9-9. Also included in UCM 9-1200 is a file system filter driver 9-1220, in one embodiment of the present invention.

File system filter driver 9-1220 can, in one embodiment, be an upper level and/or lower level filter for the individual bus devices within client computer system 9-210, e.g., media storage device drive 9-1112 of Figures 9-10, 9-11, 9-13, 9-14, and 9-15. File system filter driver 9-1220 is enabled to hook onto access to a media storage device drive 9-1112, e.g., a CD drive, and intercept data reads associated with accessing the content on a media storage device, e.g., media storage device 9-999.

File system filter driver 9-1220 includes a decrypter 9-1221 for providing decryption of encryptions applied to encrypted content, e.g., encryptions 9-2351 to 9-235N applied to encryptions 9-1351 to 9-135N of media content 9-2001 to 9-200N of Figure 9-13, in one embodiment of the present invention. Decrypter 9-1221 can provide dynamic decryption of encryptions applied to encrypted media content on a media storage device 9-999 as the content, e.g., 9-2001 to 9-200N, is accessed and read by media storage device drive 9-1112.

Still referring to Figure 9-12, UCM 9-1200 also includes a secure media player 9-1210. Secure media player 9-1210 can be, in one embodiment, similar to custom media device 9-310, an emulation of the custom media device driver 9-307, as described herein with reference to Figures 9-3 and 9-5B to 9-5D. Alternatively, secure media player 9-1210 may be an alternative media player having controlling properties analogous to custom media device 9-310. Secure media player 9-1210 includes a decrypter 9-1211 for decrypting encryption applied to each instance of media content disposed on a media storage device 9-999, e.g., encryptions 9-1351 to 9-135N applied to media content 9-2001 to 9-200N of Figure 9-13, respectively. Secure media player 9-1210 also includes a watermark 9-1212 for watermarking the outgoing data stream. In one embodiment, watermark 9-1212 operates concurrent with secure media player 9-1210 and during player 9-1210's rendering of the content, watermark 9-1212 will attach a serial number, e.g., serial number 9-1380 of Figure 9-13, associated with each media storage device 9-999 onto the outgoing data stream.

Figure 9-13 is a block diagram of contents and components that may be disposed on a media storage device, e.g., 9-999, in accordance with embodiments of the present invention. Figure 9-13 is shown with multiple instances of content, e.g. media content 9-2001 to 9-200N, disposed thereon. Media content 9-2001 to 9-200N may be, but are not limited to, movies, audio tracks, beta software, documents, literary works, etc. It is noted that any digital media can be disposed on a media storage device 9-999 or on a plurality of media storage devices 9-999.

Media storage device 9-999 of Figure 9-13 is analogous to media storage device 9-999 of Figures 9-10 and 9-11, 9-14, and 9-15. In one embodiment of the present invention, media storage device 9-999 is configured for utilization in conjunction with demonstration and/or pre-release content.

Media storage device 9-999 of Figure 9-13 is shown to have disposed thereon a UCM (usage compliance mechanism) 9-1200 for controlling presentation of content, e.g., media content 9-2001 to 9-200N, disposed on media storage device 9-999. The UCM 9-1200 in Figures 9-13, 9-14, and 9-15, is analogous to the UCM 9-1200 described herein with reference to Figure 9-12. It is noted that autorun protocol 9-910 of UCM 9-1200 is, in one embodiment, disposed on media storage device 9-999 in a non-encrypted form.

Also shown on media storage device 9-999 is a unique identifier for providing a unique identification of the media storage device, e.g., serial number 9-1380, in one embodiment of the present invention. Serial number 9-1380 may be, but is not limited to, nearly any distinguishable identifying type of indicator, e.g., a randomly generated number, a sequential number, a combination of numbers and alphanumeric characters, and the like.

Advantageously, by disposing a unique identifier on a media storage device 9-999, e.g. a serial number 9-1380, this enables close association of the content disposed thereon, e.g., media content 9-2001 to 9-200N, with the anticipated recipient of the media storage device, e.g., a movie critic, a music critic, an academy award member, a software beta tester, etc. Therefore, by closely associating a

media storage device, e.g., 9-999, with an anticipated recipient, e.g., the user of computer system 9-210, embodiments of the present invention can prevent unauthorized persons from experiencing content on a media storage device, as described herein with reference to Figures 9-3, 9-4, 9-7A to 9-7C, and 9-8. Further advantageous is that by having a unique identifier for each media storage device 9-999, embodiments also provide security at the media storage device mastering level. This means that an employee working at a mastering facility who unlawfully purloins a copy of the media storage device may still be able to copy the contents and turn those copies into bootleg (unauthorized versions) copies of the media storage device in an attempt to flood the market. However, by virtue of each media storage device 9-999 having a unique identifier, and each media storage device 9-999 is associated with its intended recipient, persons not associated with a particular media storage device 9-999 will be unable to experience the content thereon. While the market may still be flooded with bootleg copies, those that acquire a bootleg copy of a media storage device 9-999, in accordance with the present invention, will not be able to experience the content thereon, thereby possibly causing the public to be less receptive to the idea of an inexpensive bootleg copy of something that they cannot use.

In one embodiment, media storage device 9-999 may be distributed to its intended recipients in a variety of ways. Ways to distribute media storage device 9-999 to its intended recipients can include, but is not limited to, postal delivery methods, e.g., the United States Postal Service, parcel delivery services such UPS (United Parcel Service) and/or Federal Express, courier delivery services, and the

like. In another embodiment, the intended recipient of a media storage device 9-999 may be required to physically pick up device 9-999 from a distribution point.

Also shown on media storage device 9-999 are multiple instances of content, e.g., media content 9-2001 to 9-200N, in one embodiment of the present invention. Media content 9-2001 to 9-200N can be any type of digital media content, including, but not limited to, audio, video, multimedia, graphics, information, data, software programs, etc.

Still referring to Figure 9-13, in one embodiment of the present invention, each instance of media content 9-2001 to 9-200N is subject to a first encryption, e.g., encryptions 9-1351 to 9-135N, respectively. In one embodiment, a first decryption key for each encryption, e.g., encryptions 9-1351 to 9-135N, may be stored in a server, e.g., web server 9-250 and/or content server 9-251 of Figures 9-2, 9-4, 9-10, and 9-14. In one embodiment, secure media player 9-1210 can utilize decrypter 9-1211 and the decryption key stored on web server 9-250 and/or content server 9-251 and decrypt encryptions 9-1351 to 9-135N during rendering of the content. It is noted that secure media player 9-1210 is communicatively coupled with web server 9-250 and/or content server 9-251 during rendering and presentation of the content disposed on media storage device 9-999.

Additionally, media content 9-2001 to 9-200N having a first encryption applied thereto, e.g., encryptions 9-1351 to 9-135N, are each subject to a second encryption, e.g., encryptions 9-2351 to 9-235N, respectively, prior to disposal of media content 9-2001 to 9-200N on a media storage device 9-999. In one embodiment, a second

decryption key to decrypt encryptions 9-2351 to 9-235N may be stored in a server, e.g., web server 9-250 and/or content server 9-251 of Figures 9-2, 9-4, 9-10, and 9-14. In one embodiment, file system filter driver 9-1220 can utilize decrypter 9-1221 and the second decryption key stored on web server 9-250 and/or content server 9-251 and decrypt encryptions 9-2351 to 9-235N during reading of the content on media storage device 9-999 by media storage device drive 9-1112. It is noted that file system filter driver 9-1220 is communicatively coupled with web server 9-250 and/or content server 9-251 during rendering and presentation.

In one embodiment, encryptions 9-1351 to 9-135N can be less computationally intensive encryptions than encryptions 9-2351 to 9-235N. Alternatively, in one embodiment, encryptions 9-1351 to 9-135N can be more computationally intensive than encryptions 9-2351 to 9-235N.

There are many available encryption methods that can be implemented as encryptions 9-1351 to 9-135N and/or encryptions 9-2351 to 9-235N. Examples of encryptions that may be implemented as encryptions 9-1351 to 9-135N and/or 9-2351 to 9-235N can include, but are not limited to, triple DES, AES, Blowfish, and numerous others. In one embodiment, encryptions 9-1351 to 9-135N and/or 9-2351 to 9-235N can each be comprised of a series and/or a mixture of encryptions. A differing encryption, e.g., a plurality of randomly generated encryptions, can be implemented for each instance of media content on a media storage device, rather than using one format. In one embodiment, numerous variations of Blowfish are utilized to provide the desired encryptions.

Advantageously, by utilizing multiple differing encryptions for each instance of media content, e.g., 9-2001 to 9-200N, if a person/hacker attempts to gain access to the content by breaking encryption applied to a media content, e.g., second encryption 9-2352 applied to encrypted media content 9-2002, and succeeds, they have simply broken the second encryption for media content 9-2002. However, the remaining encryption 9-1352 remains unbroken by virtue of the differing encryptions. Therefore, the person/hacker would have to perform the entire encryption breaking process again to access media content 9-2002 on media storage device 9-999. Thus, after spending non-trivial amounts of time breaking two differing encryptions applied to an instance of media content, e.g., content 9-2002, the remaining content on media storage device 9-999 is still encrypted, each with its own differing multiple encryption.

Figure 9-14 is a block diagram of a communicative environment 9-1400 for controlling presentation of media content disposed on a media storage device. Included in communicative environment 9-1400 is a media storage device drive 9-1112 coupled with a client computer system 9-210 via a data/address bus 9-110. Client computer system 9-210 is coupled with web server 9-250 and/or content server 9-251 via Internet 9-201. A media storage device 9-999, upon which a usage compliance mechanism 9-1200 may be disposed, is inserted in media storage device drive 9-1112. Autorun protocol component 9-910 detects the insertion and automatically invokes installation of CCM 9-300, file system filter driver 9-1220 and secure media player 9-1210 from media storage device 9-999 into client computer system 9-210. Subsequent to installation, UCM 9-1200 initiates a dynamic update with web server 9-250 and/or content server 9-251, via Internet 9-201, as described

herein with reference to Figures 9-3, 9-4, and 9-7A to 9-7C, thereby controlling the integrity of the software. Additionally, by conferring with servers 9-250 and/or 9-251 via Internet 9-201 online, the UCM 9-1200 software version on media storage device 9-999 and installed on client computer system 9-210 can be updated when circumventions occur and kept current from platform to platform.

Advantageously, the monitoring mechanism of agent program 9-304 enables constant morphing of the version of CCM 9-300 disposed on media storage device 9-999 by communicating with server 9-250 and/or 9-251 and utilizing the dynamic update capabilities of global network 9-800 to readily update that which has been installed on client computer system 9-210, via media storage device 9-999.

In one embodiment, the installation is performed clandestine with respect to the recipient of media storage device 9-999 and is initiated by inserting media storage device 9-999 into an appropriate media storage device drive, e.g. a magnetic/optical disk drive or alternative device drive coupled with client system 9-210. Portions of UCM 9-1200 determine if the recipient is registered with web server 9-250 and/or content server 9-251. If the recipient is not registered with servers 9-250 and/or 9-251, as described herein with reference to Figure 9-4 and Figures 9-7A to 9-7C, and Figure 9-8, portions of UCM 9-1200 initiates an installation process as described herein with reference to Figures 9-3, 9-4, 9-7A to 9-7C, and 9-11.

If computer system 9-210 is registered with servers 9-250 and/or 9-251, UCM 9-1200 can initiate an update process with web server 9-250 and/or content server 9-251 to readily include updates that have been invoked subsequent to distribution of

media storage device 9-999. By virtue of the dynamic update capabilities of UCM 9-300, regardless of the version of CCM 9-300 on media storage device 9-999, UCM 9-1200 provides compliance with copyright restrictions and licensing agreements applicable to the media content on media storage device 9-999, e.g., media content 9-2001 to 9-200N. Advantageously, enabling dynamic adaptability of UCM 9-1200 provides for continued interoperability with new and updated operating systems, advancements in electronic technology, communication technologies and protocols, and the like, ensuring the effectiveness of UCM 9-1200 into the future.

In another embodiment, if the user is a registered user with global delivery system 9-800, UCM 9-1200 can detect which version is most current. Accordingly, when the version existing on client system 9-210 is more current than the version (for install) on media storage device 9-999, UCM 9-1200 can bypass the install process and present the contents contained on media storage device 9-999 to the user for them to experience.

Further advantageously, this technology is backward compatible with media storage device drives manufactured subsequent to 1982. Additionally, UCM 9-1200 is compatible with media storage devices having a copyright indicator bit disposed thereon. The copyright indicator bit has been included on all CDs released since 1982.

In the present embodiment of Figure 9-14, each instance of media content is encrypted on media storage device 9-999, as described herein with reference to Figure 9-13. However, most home players and/or stand alone media playing devices

rarely include a decryption mechanism, and to experience the music on a home machine, the music is conventionally not encrypted. Accordingly, media storage device 9-999, in its present embodiment, may not be operable on a home and/or stand alone media playing device.

In one embodiment, UCM 9-1200 can invoke its own proprietary player, e.g., secure media player 9-1210, as described with reference to custom media device 9-310 of Figure 9-3, thus enabling increased control of copyright restrictions and/or licensing agreements applicable to the media content. By invoking a secure media player 1210, UCM 9-1200 enables user experience of media content while providing protection against unauthorized presentation or reproduction of the media disposed on media storage device 9-999.

Still referring to Figure 9-14, in one embodiment, the media content, e.g., media content 9-2001 to 9-200N, and UCM 9-1200 disposed on a media storage device 9-999 are encrypted, with the exception of autorun protocol 9-910, as described above. In one embodiment of the present invention, UCM 9-1200 is encrypted differently than media content 9-2001 to 9-200N, thereby preventing the cracking of one encryption from being utilized on another encryption. This implementation is particularly advantageous for demonstration (demo) versions of media files, beta test versions, and the like that may be disposed on media storage device 9-999. It is noted that the present embodiment is operable in an online environment, meaning that client computer system 9-210 is communicatively coupled with web server 9-250 and/or content server 9-251 to enable a user experience of the content on a demo version of media storage device 9-999. In this

implementation, UCM 9-1200 allows for specific plays for specific users, which can be controlled via a network, e.g., network 9-1400 of Figure 9-14, and server 9-250 and/or 9-251.

In the present embodiment, UCM 9-1200 can be implemented for demonstration and/or pre-release protection of content disposed on a media storage device 9-999. However, content disposed on media storage device 9-999 can also be commercially released content, e.g., audio, video, software, and the like. In this embodiment, sophisticated encryption technology, e.g., Blowfish, is utilized to encrypt media content 9-2001 to 9-200N on media storage device 9-999 with an associated decrypter key located on web server 9-250 and/or content server 9-251. In one embodiment, a plurality of encryptions are applied to media content 9-2001 to 9-200N and a plurality of decrypter keys are stored on web server 9-250 and/or content server 9-251. Decryption can be performed using an associated decryption key in conjunction with a secure media player 9-1210 and file system filter driver 9-1220 installed on computer system 9-210 via media storage device 9-999.

Still with reference to Figure 9-14, the content of media storage device 9-999, e.g., media content 9-2001 to 9-200N is encrypted, using various levels of encryption to provide protection levels commensurate with copyright holders desires and required protection. For example, media storage device 9-999 is delivered to a user or critic for the purposes of review, the user inserts media storage device 9-999 into the appropriate storage device reader or connector coupled with the recipient's computer, and autorun protocol 9-910 initiates UCM 9-1200 install of CCM 9-300, file system filter driver 9-1220, secure media player 9-1210 on client system 9-210 in

a manner clandestine to the user. Once installed, UCM 9-1200 initiates a communication session with web server 9-250/content server 9-251, where content server 9-251 can provide authorization for the user to experience the media on media storage device 9-999.

Accordingly, if the user, to whom demo media storage device 9-999 had been released, had demo media storage device 9-999 stolen, or if the user allowed alternative parties try to experience the content of media storage device 9-999, the unauthorized party would have to try to crack the encryption keys and the encryption of the actual content of media storage device 9-999, consuming non-trivial amounts of time.

Thus, UCM 9-1200 is able to control which recipients receive authorization to experience the media content on media storage device 9-999, how many times the recipient may experience the media, and UCM 9-1200 may also define a period of time until the media content may no longer be accessible. This may enable copyright holders to release the media content on an authorized media storage device, e.g., 9-999, prior to pirated copies flooding the market.

Still referring to Figure 9-14, accordingly, a media storage device 9-999 may be configured such that a first user may get a copy, a second user may get a copy, and if it is known that the second user will share the demo with a third and a fourth user, then the known users would be enabled to experience the media.

Advantageously, by virtue of defining which users can access and experience the media, any unauthorized sharing of the media by one of the authorized users can be

readily detected, and further sharing or experiencing of the media may be halted. Additionally, because the authorized user shared the media in an unauthorized manner, in a worse case scenario, criminal charges could be filed against that user.

It is noted that placing UCM 9-1200 on a media storage device, e.g., 9-999, so as to enable installation of CCM 9-300 on client system 9-210 is one manner in which CCM 9-300 can be installed on client system 9-210. An alternative manner in which CCM 9-300 can be installed on client computer system 9-210 is through "cross-pollination." For example, webcasters broadcast the media file to the user. The media file has a CCM 9-300 coupled with the media file, and upon downloading the media file onto client computer system 9-210, embodiments of the present invention enable the installation of CCM 9-300 onto client computer system 9-210. In another manner, CCM 9-300 is incorporated into and becomes part of an operating system operational on client system 9-210. Alternatively, laws are passed that mandate the inclusion of CCM 9-300 on each client computer system 9-210.

Figure 9-15 is an exemplary logic/bit path block diagram 9-1500 of a client computer system, e.g., 9-210, configured with a usage compliance mechanism 9-1200 for controlling presentation of content on a media storage device 9-999, in accordance with one embodiment of the present invention. Usage compliance mechanism 9-1200 of Figure 9-15 is analogous to usage compliance mechanism 9-1200 of Figure 9-12. Therefore, CCM 9-300 of usage compliance mechanism 9-1200 is analogous to a copyright compliance mechanism 9-300 coupled with and installed on a client computer system, e.g., 9-210, as described herein with reference

to Figures 9-3, 9-4, 9-5A to 9-5D, 9-6A, 9-7A to 9-7C, 9-8, 9-9, 9-10, 9-11 9-14, 9-15, and 9-16.

Diagram 9-1500 of Figure 9-15 includes a media storage device drive 9-1112 coupled with a media storage device file system driver 9-1114 via line 9-1571. Media storage device drive file system driver 9-1114 enables an operating system, e.g., Windows by Microsoft, Apple by Apple, Linux by Linux, etc., on a client computer system, e.g., client 9-210, to recognize and control the media storage device drive, e.g., drive 9-1112. Coupled to media storage device drive file system driver 9-1114 is file system filter driver 9-1220, via line 9-1572. Coupled to file system filter driver 9-1220 is a secure media player 9-1210 via line 9-1573. Coupled with secure media player 9-1210 are an operating system media subsystem 9-503 via line 9-1577 and a media hardware output device 9-1370 via line 9-1574. UCM 9-1200 is coupled with operating system media subsystem 9-503 via line 9-1576.

Media storage device drive 9-1112 of Figure 9-15 is analogous to media storage device drive 9-1112 of Figures 9-11 and 9-15. Media storage device drive 9-1112 is configured to receive a media storage device 9-999, where media storage device 9-999 is appropriate for drive 9-1112. In one embodiment, drive 9-1112 may be a CD drive and accordingly, media storage device 9-999 would be a CD. In another embodiment, drive 9-1112 may be a DVD drive and accordingly, media storage device 9-999 would be a DVD, and so on. Therefore, media storage device drive 9-1112 can, when so configured, receive any media storage device 9-999 upon which data or content may be disposed.

File system filter driver 9-1220 can be an upper level and/or lower level filter for the individual bus devices within client computer system 9-210, e.g., media storage device drive 9-1112, and is analogous to file system filter driver 9-1220 of Figure 9-12. File system filter driver 9-1220 is able to hook onto access to a media storage device drive 9-1112, e.g., a CD drive, and intercept data reads associated with accessing the content, e.g., media content 9-2001 to 9-200N, on a media storage device, e.g., media storage device 9-999. File system filter driver 9-1220 is also enabled, via decrypter 9-1221 and a decrypter key on servers 9-250 and/or 9-251, to provide dynamic decryption of encrypted media content on a media storage device 9-999 as the content is accessed and read by media storage device drive 9-1112.

By virtue of file system filter driver 9-1220 operating at a file system level instead of operating at a device drive class level, e.g., a CD class level, it is able to recognize which files are being accessed from media storage device 9-999 for a particular operation. Advantageously, this obviates the need for a file system to be implemented within a driver for determining whether data that is being read needs decrypting.

Still referring to Figure 9-15, secure media player 9-1210 is analogous to secure media player 9-1210 of Figure 9-11. Secure media player 9-1210 can, in one embodiment, be a custom media device 9-310 emulated by a custom media device driver 9-307, as described herein with reference to Figure 9-3. In another embodiment, secure media player 9-1210 can be a proprietary player configured for utilization with demonstration and/or pre-release content disposed on a media

storage device, e.g., media content 9-2001 to 9-200N. Other authorized media players may also be used to present media content on a media storage device 9-999, provided the other media players can comply with usage restrictions and licensing agreements applicable to the media content and provided by secure media player 9-1210.

Media hardware output device 9-1370 is an appropriate output device for the media content on media storage device 9-999. If media content 9-2001 to 9-200N are audio tracks or songs, then output device 9-1370 is an audio or sound card for outputting music via speakers. Alternatively, if media content 9-2001 to 9-200N are video tracks, movies, literary works, software programs, etc., then output device 9-1370 is a graphics card for outputting movies, text, and the like via a display device, e.g., display device 9-105 of Figure 9-1.

Continuing with Figure 9-15, a media storage device 9-999 is received by a media storage device drive 9-1112. Autorun protocol 9-910 initiates a process to determine the presence of a usage compliance mechanism 9-1200 and a secure media player 9-1210 operable on computer system 9-210. If either and/or both usage compliance mechanism 9-1200 and secure media player 9-1210 are not present on computer system 9-210, autorun protocol initiates installation of the components, as described herein with reference to Figures 9-3, 9-4, 9-5A to 9-5D, 9-6, 9-7A to 9-7C, 9-8 to 9-16. If UCM 9-1200 and secure media player 9-1210 are both present, autorun 9-910 bypasses the installation thereof. Media storage device file system driver 9-1114 accesses the content on media storage device, e.g., media content 9-2001 to 9-200N, and reads the data.

File system filter driver 9-1220 intercepts the read operation being performed by driver 9-1114 and dynamically decrypts a second encryption applied to media content 9-2001 to 9-200N, e.g., encryptions 9-2351 to 9-235N of Figure 9-13, via decrypter 9-1221 and a second decryption key stored on and retrieved from servers 9-250 and/or 9-251. In one embodiment, if file system filter driver 9-1220 is not communicatively coupled with server 9-250 and/or 9-251, thereby enabled to retrieve the second decryption key, presentation of the content on a media storage device 9-999 is not permitted.

Continuing with Figure 9-15, subsequent to second encryptions 9-2351 to 9-235N being decrypted, media content 9-2001 to 9-200N, still encrypted with a first encryption, e.g., encryptions 9-1351 to 9-135N, respectively, is output to secure media player 9-1210 via line 9-1573. Secure media player 9-1210 in conjunction with UCM 9-1200 communicates with server 9-250 and/or 9-251 and determines if computer system 9-210 and the user thereof, are authorized to experience media content 9-2001 to 9-200N. If system 9-210 and the user thereof are authorized to experience media content 9-2001 to 9-200N, secure media player 9-1210 commences to render the media content for presentation via media hardware output device 9-1370.

Concurrent with rendering media content 9-2001 to 9-200N, secure media player 9-1210, can, in one embodiment, communicate with server 9-250 and/or 9-251 and retrieve the decryption key associated with each encryption, e.g., 9-1351 to 9-135N, and with decrypter 9-1211 of Figure 9-12, dynamically decrypt each

instance of media content, e.g., 9-2001 to 9-200N, as the content is being rendered and output via line 9-1574 to media hardware output device 9-1370.

Because rendered content is vulnerable to capture and/or imaging, and thus becoming subject to ripping, burning, copying, and the like, secure media player 9-1210 can watermark, via watermarker 9-1212 the outgoing data stream that is output to media hardware output device 9-1370 via line 9-1574. In one embodiment, utilizing watermarker 9-1212, the outgoing data stream is watermarked concurrent with the rendering performed by secure media player 9-1210. Further, secure media player 9-1210 attaches a unique identifier with each rendered media content 9-2001 to 9-200N. In one embodiment, serial number 9-1380 is attached to each media content 9-2001 to 9-200N, as it is being rendered and output to media hardware output device 9-1370. In this manner, if the rendered content being output is somehow captured, imaged, etc., by virtue of the association of serial number 9-1380 with media storage device 9-999 and the media content disposed thereon, e.g., content 9-2001 to 9-200N, and the computer system 9-210 with which the recipient of media storage device 9-999 is associated, unauthorized presentation and reproduction of the media content is prevented.

Figure 9-16 is a flowchart 9-1600 of computer implemented steps performed in accordance with one embodiment of the present invention for controlling presentation of media content disposed on a media storage device. Flowchart 9-1600 includes processes of the present invention, which, in one embodiment, are carried out by processors and electrical components under control of computer readable and computer executable instructions. The computer readable and

computer executable instructions reside, for example, in data storage features such as computer usable volatile memory 9-104 and/or computer usable non-volatile memory 9-103 of Figure 9-1. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart 9-1600, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in Figure 9-16. Within the present embodiment, it should be appreciated that the steps of flowchart 9-1600 may be performed by software, by hardware or by any combination of software and hardware.

It is noted that flowchart 9-1600 is described in conjunction with Figures 9-2, 9-3, 9-4, 9-5A to 9-5D, 9-6, 9-7A to 9-7C, 9-8 to 9-15 to more fully describe the operation of the present embodiment. In step 9-1610, an autorun mechanism disposed on a media storage device 9-999, e.g., autorun protocol 9-910, is activated in response to a computer system 9-210 receiving media storage device 9-999 in an appropriate device drive, e.g., media storage device drive 9-1112.

In step 9-1612 of Figure 9-16, a monitoring program within UCM 9-1200 disposed on media storage device 9-999 determines if a usage compliance mechanism, e.g., UCM 9-1200, is installed on the computer system which received media storage device 9-999, e.g., computer system 9-210. In one embodiment, agent programs 9-304 may perform the determination. However, in another embodiment, combinations of components of a CCM 9-300, as described herein with reference to Figure 9-3, 9-4, 9-7A to 9-7C may be utilized.

If UCM 9-1200 is not present on computer system 9-210, the present method proceeds to step 9-1611. Step 9-1611 installs a usage compliance mechanism on computer system 9-210, as described herein with reference to Figures 9-3, 9-4, 9-5A to 9-5D, 9-6, 9-7A to 9-7C, and 9-8 to 9-15. Alternatively, if UCM 9-1200 is present on computer system 9-210, the present method proceeds to step 9-1614.

In step 9-1614, a monitoring mechanism within UCM 9-1200 disposed on media storage device 9-999 determines if a secure media player 9-1210 is present and operable on computer system 9-210. In one embodiment, agent program 9-304 of CCM 9-300 of Figure 9-3 can provide the determination.

If a secure media player 9-1210 is not present and operable on computer system 9-210, the present method proceeds to step 9-1611. Step 9-1611 installs a secure media player 9-1210 on computer system 9-210, as described herein with reference to Figures 9-3, 9-4, 9-5A to 9-5D, 9-6, 9-7A to 9-7C, and 9-8 to 9-15. It is noted that if usage compliance mechanism 9-1200 and/or secure media player 9-1210 cannot be properly installed on computer system 9-210, the present method proceeds to step 9-1622, which ends the session and which prevents computer system 9-210 from presenting the content on media storage device 9-999. Alternatively, if a secure media player 9-1210 is present and operable on computer system 9-210, the present method proceeds to step 9-1616.

In step 9-1616, UCM 9-1200 communicates with servers 9-250 and/or 9-251 in networks 9-200, 9-400, 9-1000, and/or 9-1400 and determines whether computer system 9-210 and the user thereof are authorized to experience media content 9-

2001 to 9-200N on media storage device 9-999, as described herein with reference to Figures 9-3, 9-4, steps 9-704 to 9-708 of Figure 9-7A and 9-8 to 9-15. If computer system 9-210 and the user thereof are not authorized to experience the content on media storage device 9-999, the present method proceeds to step 9-1615.

Alternatively, if computer system 9-210 and the user thereof are authorized to experience the content on media storage device 9-999, the present method proceeds to step 9-1618.

In step 9-1618 of Figure 9-16, UCM 9-1200 determines if secure media player 9-1210, usage compliance mechanism 9-1200 and computer system 9-210 are all communicatively coupled with networks 9-200, 9-400, 9-1000, and/or 9-1400. If one or more of the conditions are not met, the present method proceeds to step 9-1615. Alternatively, if secure media player 9-1210, usage compliance mechanism 9-1200, and computer system are all communicatively coupled with networks 9-200, 9-400, 9-1000, and/or 9-1400, the present method proceeds to step 9-1620.

Step 9-1615 of Figure 9-16 prevents presentation of content on a media storage device, e.g., media content 9-2001 to 9-200N on media storage device 9-999, to the user of computer system 9-210. Alternatively, computer system 9-210 and the user thereof may communicate with networks 9-200, 9-400, 9-1000, and/or 9-1400 and attempt to establish credentials and/or to re-establish a communicative coupling with networks 9-200, 9-400, 9-1000, and/or 9-1400, that would allow presentation of the content, as described herein with reference to Figure 9-3, 9-4, and steps 9-704 to 9-708 of Figure 9-7A.

In step 9-1620, media content 9-2001 to 9-200N on media storage device 9-999 is read by media storage device drive 9-1112. File system filter driver 9-1220 intercepts the read operation being performed by media storage device file system driver 9-1114 and dynamically decrypts a second encryption applied to media content 9-2001 to 9-200N, e.g., encryptions 9-2351 to 9-235N of Figure 9-13, via decrypter 9-1221 and a second decryption key stored on and retrieved from servers 9-250 and/or 9-251. It is noted that if file system filter driver 9-1220 is not communicatively coupled with server 9-250 and/or 9-251, enabling retrieval of the second decryption key, presentation of the content on a media storage device 9-999 is not permitted.

Continuing with step 9-1620 of Figure 9-16, subsequent to second encryption 9-2351 to 9-235N being decrypted, media content 9-2001 to 9-200N, still encrypted with a first encryption, e.g., encryptions 9-1351 to 9-135N, respectively, is output to secure media player 9-1210. In one embodiment, secure media player 9-1210, in conjunction with UCM 9-1200, communicates with server 9-250 and/or 9-251 and commences to render the media content for presentation via media hardware output device 9-1370.

Concurrent with rendering media content 9-2001 to 9-200N, secure media player 9-1210, can, in one embodiment, communicate with server 9-250 and/or 9-251 and retrieve the decryption key associated with each encryption, e.g., 9-1351 to 9-135N, and with decrypter 9-1211 of Figure 9-12, dynamically decrypt each instance of media content, e.g., 9-2001 to 9-200N, as the content is being rendered and output the rendered content to media hardware output device 9-1370.

Because rendered content is vulnerable to capture and/or imaging, and thus becoming subject to ripping, burning, copying, and the like, secure media player 9-1210 can watermark, via watermarker 9-1212, the outgoing data stream that is output to media hardware output device 9-1370. In one embodiment, the outgoing data stream is watermarked concurrent with the rendering performed by secure media player 9-1210. Further, secure media player 9-1210 attaches a unique identifier with each rendered media content 9-2001 to 9-200N. In one embodiment, serial number 9-1380 is attached to each media content 9-2001 to 9-200N, as it is being rendered and output to media hardware output device 9-1370. In this manner, if the rendered content being output is somehow captured, imaged, etc., by virtue of the association of serial number 9-1380 with media storage device 9-999 and the media content disposed thereon, e.g., content 9-2001 to 9-200N, and the computer system 9-210 with which the recipient of media storage device 9-999 is associated, unauthorized presentation and reproduction of the media content is prevented.

A method of controlling presentation of content on a media storage device is described. The method is comprised of verifying presence of a content presentation mechanism and a usage compliance mechanism on a computer system operated by a recipient to whom the media storage device is distributed. The usage compliance mechanism includes a file system filter driver for controlling data reads associated with the content. The present method further includes permitting the recipient to experience the content via the computer system provided the usage compliance mechanism is present on the computer system and the computer system is communicatively coupled with a network and wherein a server in the network

authorizes the recipient to experience the content. The present method further includes presenting the content to the recipient via the content presentation mechanism. The content presentation mechanism is communicatively coupled with the usage compliance mechanism. The content presentation mechanism is enabled to present the content provided the content presentation mechanism is communicatively coupled with the server.

The foregoing disclosure regarding specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

The Applicant reserves the right to claim or disclaim now or in the future any feature, combination of features, or subcombination of features that is disclosed herein.

All of the numerical and quantitative measurements set forth in this application (including in the description, claims, abstract, drawings, and any appendices) are approximations.

The invention illustratively disclosed or claimed herein suitably may be practiced in the absence of any element which is not specifically disclosed or claimed herein. Thus, the invention may comprise, consist of, or consist essentially of the elements disclosed or claimed herein.

The following claims are entitled to the broadest possible scope consistent with this application. The claims shall not necessarily be limited to the preferred embodiments or to the embodiments shown in the examples.

All U.S. patents, prior filed patent applications (including U.S. Patent Application Nos. 60/323,468, 60/379,979, 60/378,011, 10/218,241, 10/235,293, 10/304,390, 10/325,243, 10/364,463, and 60/451,231; a U.S. patent application entitled "METHOD AND SYSTEM FOR CONTROLLING ACCESS OF MEDIA ON A MEDIA STORAGE DEVICE" (filed 5 May 2003); and a U.S. patent application entitled "METHOD AND SYSTEM FOR CONTROLLING PRESENTATION OF MEDIA ON A MEDIA STORAGE DEVICE" (filed 5 May 2003)), and any other documents and printed matter cited or referred to in this application are incorporated in their entirety herein by this reference.

CLAIMS

What is claimed is:

1. A method for providing media content from a source database to a computer, said method comprising:

- transmitting via a communication network a first request for a media content list from said computer;
- transmitting to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request;
- transmitting a second request for delivery of a media content together with said unique identification, in response to receiving said media content list;
- transmitting to said computer via said communication network an access key together with an address for said source database containing said media content, in response to said unique identification of said second request being valid;
- transmitting a third request together with said access key, in response to receiving said access key together with said address of said source database; and
- transmitting said media content to said computer, in response to said access key being valid.

2. The method according to Claim 1, wherein said method prevents unauthorized access to said source database.

3. A computer readable medium having computer readable code embodied therein for causing a system to perform:

- transmitting via a communication network a first request for a media content list from a computer;
- transmitting to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request;

transmitting a second request for delivery of a media content of said media content list together with said unique identification, in response to receiving said media content list;

transmitting to said computer via said communication network an access key together with an address for said media content, in response to said unique identification of said second request being valid;

transmitting a third request together with said access key, in response to receiving said access key together with said address of said media content;

transmitting said media content to said computer, in response to said access key being valid.

4. A system for providing a media content stored by a content server to a computer, said system comprising:

means for transferring via a communication network a first request for a media content list from said computer;

means for transferring to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request;

means for transferring a second request for delivery of a media content of said media content list together with said unique identification, in response to receiving said media content list;

means for transferring to said computer via said communication network a time sensitive access key together with an address of said content server containing said media content, in response to said unique identification of said second request being valid;

means for transferring a third request together with said time sensitive access key to said content server, in response to receiving said time sensitive access key together with said address of said content server; and

means for transferring said media content to said computer, in response to said time sensitive access key being valid.

5. The system according to Claim 4, wherein said system prevents unauthorized access to said media content stored by said content server.

6. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, or the system according to Claims 4 or 5, wherein said communication network comprises the Internet.

7. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, or the system according to Claims 4 or 5, wherein said communication network is selected from a local area network (LAN), a wide area network (WAN) or the Internet.

8. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, or the system according to Claims 4 or 5, wherein said computer is selected from a computer system, a desktop computer, a laptop computer, and a portable computing device.

9. The method according to Claims 1 or 2, or the computer readable medium according to Claim 3, wherein said access key is a time sensitive access key.

10. The method or computer readable medium according to Claim 9 or the system according to Claims 4 or 5, wherein said time sensitive key becomes obsolete after a defined amount of time.

11. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, or the system according to Claims 4 or 5, wherein said media content is selected from an audio clip, a song, a video clip, a picture, a graphics picture, and a multimedia clip.

12. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, or the system according to Claims 4 or 5, wherein said media content comprises audio, music, video, a picture, graphics, or multimedia.

13. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, the system according to Claims 4 or 5, or the method, computer readable medium, or system according to Claim 12, wherein said media content comprises a digital watermark or an embedded key.

14. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, the system according to Claims 4 or 5, wherein said transmitting a second request for delivery of said media content is performed by an application.

15. The method, computer readable medium or system according to Claim 14, wherein said application is a software application.

16. The method or computer readable medium according to Claims 14 or 15 further comprising:
determining whether said application is valid.

17. The system according to Claims 4 or 5, wherein said transmitting a second request for delivery of said media content is performed by an application and said system further comprises:
means for determining if said application is valid.

18. The method or computer readable medium according to Claim 16 or the system according to Claim 17 wherein said application is valid when said application is a media player application.

19. The method according to Claims 1 or 2, the computer readable medium according to Claim 3, the system according to Claims 4 or 5, wherein said address is changed subsequently after said transmitting of said address to said computer.

20. A system for providing media content, said system comprising:
a media content library having a plurality of media content pieces;

a global media content delivery network having a plurality of content sources, each content source of said plurality of content sources stores a portion of said media content library; and

a security system coupled to said plurality of content sources for allowing delivery of a media content piece from said media content library to an authorized client device while preventing unauthorized access to said media content library stored by said plurality of content sources.

21. The system according to Claim 20, wherein said system prevents unauthorized access to said plurality of content sources.

22. A method for providing media content to a client device from a plurality of content sources, said method comprising:

transmitting a first request for a media content list of a media content library from said client device;

in response to receiving said first request, transmitting to said client device said media content list together with a unique identification;

in response to receiving said media content list, transmitting a second request for delivery of said media content together with said unique identification from said client device;

provided said unique identification is valid, transmitting to said client device an access key together with a location of a content source of said plurality of content sources containing said media content;

in response to receiving said access key together with said location of said content source, transmitting a third request together with said access key to said content source from said client device; and

provided said access key is valid, transmitting said media content to said client device.

23. The method according to Claim 22, wherein said method prevents unauthorized access to said media content.

24. A system for providing media content, said system comprising:

a media content library having a plurality of media content pieces;
a global media content delivery network having a plurality of content sources, each content source of said plurality of content sources stores a portion of said media content library; and
an authorization system coupled to said plurality of content sources for regulating delivery of a media content piece from said media content library to an authorized client device while restricting unauthorized access to said media content library stored by said plurality of content sources.

25. The system according to Claim 24, wherein said system restricts unauthorized access to said plurality of content sources.

26. The system according to any one of Claims 20, 21, 24, or 25, wherein said global media content delivery network comprises a local area network (LAN), wide area network (WAN) or the Internet.

27. The system according to Claims 24 or 25, wherein said authorized client device is a computer.

28. The system according to any one of Claims 20, 21, 24, 25, or 27, or the method according to Claims 22 or 23, wherein said client device is selected from a computer system, a desktop computer, a laptop computer, and a portable computing device.

29. The system according to any one of Claims 20, 21, 24, or 25, or the method according to Claims 22 or 23, wherein said media content piece comprises audio, music, video, a picture, graphics, or multimedia.

30. The system according to any one of Claims 20, 21, 24 or 25, or the system or method according to Claim 29, wherein said media content piece comprises a digital watermark or an embedded key.

31. The system according to any one of Claims 20, 21, 24 or 25, or the method according to Claims 22 or 23, wherein said media content is selected from an audio clip, a song, a video clip, a picture, a graphics picture, and a multimedia clip.

32. The system according to any one of Claims 20, 21, 24, or 25, or the method according to Claims 22 or 23, wherein said media content comprises a digital watermark or an embedded key.

33. The system according to Claims 20 or 21, wherein said security system for further providing an access key to said authorized client device for accessing said media content piece from a content source of said plurality of content sources.

34. The system according to Claims 24 or 25, wherein said authorization system for further providing an access key to said authorized client computer for accessing said media content piece from a content source of said plurality of content sources.

35. The system according to Claims 33 or 34, wherein said access key is a time sensitive access key that becomes obsolete after a defined amount of time.

36. The system according to Claims 20 or 21, wherein said security system for further preventing access to said media content library when a media player application operating on a client device is determined invalid.

37. The system according to Claims 20 or 21, wherein said security system for further preventing access to said media content library when an identification provided by a client device is determined invalid.

38. The system according to Claims 24 or 25, wherein said authorization system for further preventing access to said media content library when a media player application operating on a client computer is determined invalid.

39. The system according to Claims 24 or 25, wherein said authorization system for further preventing access to said media content library when an identification provided by a client computer is determined invalid.

40. The method according to Claims 22 or 23 further comprising:
determining which content source of said plurality of content sources is nearest to the location of said client device.

41. The method according to Claims 22 or 23 further comprising:
determining which content source of said plurality of content sources contains said media content.

42. The method according to Claims 22 or 23 wherein said location of said content providing source comprises an address to said media content.

43. The method according to Claim 42 wherein said address is changed after said transmitting said media content to said client device.

44. The method according to Claims 22 or 23 wherein each content source of said plurality of content sources stores a portion of said media content library.

45. The method according to Claims 22 or 23 wherein each content source of said plurality of content sources stores a copy of said media content library.

46. A method of controlling interaction of deliverable electronic media comprising:
detecting a media player application operable with a computer system, said media player application for enabling said computer system to present contents of a media file;

governing within said media player application a function that enables non-compliance with a usage restriction applicable to said media file; and

controlling the output of said media file, said controlling by a compliance mechanism coupled to said computer system, said compliance mechanism for enabling compliance with said usage restriction applicable to said media file.

47. A computer readable medium for storing computer implementable instructions, said instructions for causing a compliance mechanism to perform a method of controlling interaction of a media file, said method comprising:

discovering a media player application operable within a client computer system, said media player application for presenting contents of a media file deliverable to said client computer system;

regulating a function of said media player application that does not comply with usage restrictions applicable to said media file; and

controlling output of said media file, wherein said compliance mechanism coupled to said client computer system performs said controlling and is for enabling compliance with said usage restriction.

48. A system for media file usage restriction compliance comprising:

means for detecting a media player application operable on a client computer system and for presenting contents of a media file;

means for governing a function of said media player application that does not comply with a usage restriction applicable to a media file; and

means for controlling output of said media file, and wherein a compliance mechanism coupled to said client computer system performs said controlling and is for enabling compliance with said usage restriction applicable to said media file.

49. The method according to Claim 46, the computer readable medium according to Claim 47, or the system according to Claim 48, wherein said media file is delivered via a hypertext transfer protocol file delivery.

50. The method according to Claim 46, the computer readable medium according to Claim 47, or the system according to Claim 48, wherein said usage restriction is a copyright restriction, a licensing agreement applicable to said media file, or a license agreement pertaining to said media file.

51. The method according to Claim 46 wherein said controlling output of said media file comprises diverting a data pathway of said media player application to a controlled data pathway, and wherein said compliance mechanism controls said controlled data pathway.

52. The method according to Claim 46 further comprising delivering said media file to said computer system, said media file delivered from a server coupled with said computer system.

53. The method according to Claim 52 further comprising attaching a header to said media file prior to delivery to said computer system, said header comprising:
an indicator for indicating to said compliance mechanism that said media file originated from said server.

54. The method according to Claim 46 further comprising permitting said computer system to utilize said media player application to present contents of said media file, provided said media player application complies with said usage restriction.

55. The method according to Claim 46 further comprising installing said compliance mechanism onto said computer system, said compliance mechanism configured to perform said detecting and said disabling.

56. The method according to Claim 55 further comprising altering said compliance mechanism in response to changes in said usage restriction.

57. The method according to Claim 54 further comprising installing a custom media player application on said computer system and configured to be operable when said media player application does not comply with said usage restriction.

58. The method according to Claim 46 further comprising verifying the presence and the integrity of authorization data stored on said computer system, said verifying performed by said compliance mechanism prior to delivery of said media file to said computer system.

59. The method according to Claim 46 further comprising encrypting said media file and a header attached therewith prior to delivery of said media file to said computer system.

60. The method according to Claim 46 further comprising monitoring said media file during presentation of said contents for compliance with said usage restrictions, said monitoring performed by said compliance mechanism.

61. The computer readable medium according to Claim 47 wherein said controlling comprises redirecting a data pathway of said media player application to a controlled data pathway controlled by compliance mechanism.

62. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

initiating delivery of said media file to said client computer system from a server coupled with said client computer system.

63. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

detecting an indicator associated with said media file, said indicator for indicating said media file originated from said server.

64. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

permitting said client computer system to utilize said media player application to present said contents of said media file, provided said media player application complies with said usage restriction.

65. The computer readable medium according to Claim 64 wherein said instructions cause said compliance mechanism to perform said method further comprising:

bypassing said media player application and invoking a custom media player application coupled with said client computer system when said media player application does not comply with usage restrictions applicable to said media file, said custom media player application for presenting contents of said media file in a manner compliant with said usage restriction.

66. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

verifying the presence and integrity of authorization data stored on said client computer system.

67. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

initiating an installation of a newer version of said copyright compliance mechanism.

68. The computer readable medium according to Claim 47 wherein said instructions cause said compliance mechanism to perform said method further comprising:

monitoring said media file for compliance with said usage restrictions during presentation of said contents.

69. The system according to Claim 48 wherein said means for controlling comprises diverting a data pathway of said media player application to a controlled data pathway controlled by said compliance mechanism.

70. The system according to Claim 48 further comprising:
means for initiating delivery of said media file to said client computer system from a server coupled with said client computer system, said delivery via a hypertext transfer protocol file delivery.

71. The system according to Claim 48 further comprising:
means for permitting said client computer system to utilize said media player application when said media player application complies with said usage restriction.

72. The system according to Claim 48 further comprising:
means for deactivating said media player application when said media player application does not comply with said usage restriction.

73. The system according to Claim 48 further comprising:
means for activating a custom media player application coupled with said client computer system when said media player application is deactivated, said custom media player application for enabling said client computer system to comply with said usage restriction.

74. The system according to Claim 48 further comprising:
means for verifying the integrity of authorization data stored by said client computer system.

75. The system according to Claim 48 further comprising:
means for initiating installation of a newer version of said compliance mechanism.

76. The system according to Claim 48 further comprising:

means for detecting an indicator of a header associated with said media file, said indicator for indicating said media file originated from said server.

77. The system according to Claim 48 further comprising:
means for monitoring said media file for compliance with said usage restriction during presentation of said contents.

78. A method of preventing unauthorized recording of electronic media comprising:
activating a compliance mechanism in response to receiving media content by a client system, said compliance mechanism coupled to said client system, said client system having a media content presentation application operable thereon and coupled to said compliance mechanism;
controlling a data output path of said client computer with said compliance mechanism; and
directing said media content to a custom media device coupled to said compliance mechanism via said data output path, for selectively restricting output of said media content.

79. A computer readable medium for storing computer implementable instructions, said instructions for causing a client system to perform a method of restricting recording of media content, said method comprising:
animating a compliance mechanism coupled to said client system, said animating in response to said client system receiving media content, said client system having a media content presentation application coupled thereto and operable with said compliance mechanism;
managing an output path of said client computer with said compliance mechanism; and
governing said media content via said output path to a custom media device for selectively restricting output of said media content.

80. A system of preventing unauthorized recording of electronic media comprising:

means for activating a compliance mechanism to control a data output path of a client system, said activating in response to said client system receiving media content, said compliance mechanism coupled to said client system and operable in conjunction with a media content presentation application coupled to said client system and operable thereon; and

means for directing said media content to a custom media device via said data output path controlled by said compliance mechanism, for selectively restricting output of said media content.

81. The method according to Claim 1, the computer readable medium according to Claim 79, or the system according to Claim 80, wherein said custom media device is an emulation of a custom media driver.

82. The method according to Claim 1, the computer readable medium according to Claim 79, or the system according to Claim 80, wherein said media content is from a source coupled with said client system, said source is from the group consisting of: a network, an electronic media device, a media storage device, a media storage device inserted in a media device player, a media player application, and a media recorder application.

83. The method according to Claim 79 further comprising preventing a recording application coupled to said client system from recording said media content when said recording violates usage restriction applicable to said media content.

84. The method according to Claim 79 further comprising allowing a recording application coupled to said client system to record said media content when said recording complies with usage restrictions applicable to said media content.

85. The method according to Claim 79 further comprising restricting said client computer system to have said custom media device implemented as a default media device.

86. The method according to Claim 79 further comprising authorizing said client system to receive said media content.

87. The method according to Claim 79 further comprising accessing an indicator associated with said media content for indicating to said compliance mechanism a usage restriction applicable to said media content.

88. The method according to Claim 79 further comprising altering said compliance mechanism in response to a change in said usage restriction, said usage restriction comprising a copyright restriction or licensing agreement applicable to said media content.

89. The computer readable medium according to Claim 79 wherein said instructions cause said client system to perform said method further comprising: authorizing said client system to receive said media content.

90. The computer readable medium according to Claim 79 wherein said instructions cause said client system to perform said method further comprising: allowing a recording application coupled to said client system to record said media content file when said recording complies with a usage restriction applicable to said media content.

91. The computer readable medium according to Claim 79 wherein said instructions cause said client system to perform said method further comprising: preventing a recording application coupled to said client computer from recording said media content when said recording violates a usage restriction applicable to said media content.

92. The computer readable medium according to Claim 79 wherein said custom media device is selected as a default media device.

93. The computer readable medium according to Claim 79 wherein said instructions cause said client system to perform said method further comprising: accessing an indicator corresponding to said media content for indicating to said compliance mechanism a usage restriction applicable to said media content.

94. The computer readable medium according to Claim 79 wherein said instructions cause said client computer system to perform said method further comprising:

altering said compliance mechanism in response to changes in said usage restriction, said usage restriction a copyright restriction or licensing agreement applicable to said media content.

95. The system according to Claim 80 further comprising: means for allowing said media content to be recorded when said recording complies with usage restrictions applicable to said media content.

96. The system according to Claim 80 further comprising: means for preventing recording of said media content when said recording violates usage restriction applicable to said media content.

97. The system according to Claim 80 further comprising: means for restricting said client system to have said custom media device as a default media device.

98. The system according to Claim 80 further comprising: means for authorizing said client system to receive said media content.

99. The system according to Claim 80 further comprising: means for accessing an indicator for indicating to said compliance mechanism said usage restriction applicable to said media content, said indicator attached to said media content.

100. The system according to Claim 80 further comprising:

means for utilization of a custom media driver to emulate said custom media device.

101. The system according to Claim 80 further comprising:

means for altering said compliance mechanism in response to changes in said usage restriction, said usage restriction a copyright restriction or licensing agreement applicable to said media content.

102. A method of preventing unauthorized recording of electronic media comprising:

activating a compliance mechanism in response to a client system receiving media content, said compliance mechanism coupled to said client system, said client system having a media content presentation application operable thereon and coupled to said compliance mechanism;

controlling a data path of a kernel-mode media device driver of said client system with said compliance mechanism upon detection of a kernel streaming mechanism operable on said client system; and

directing said media content from said kernel-mode media device driver to a media device driver coupled with said compliance mechanism, via said data path, for selectively restricting output of said media content.

103. A computer readable medium for storing computer implementable instructions, said instructions for causing a client system to perform a method of restricting recording of media content, said method comprising:

animating a compliance mechanism coupled to said client system, said animating in response to said client system receiving media content, said client system having a media content presentation application coupled thereto and operable with said compliance mechanism;

managing a data path of a kernel-mode media device driver of said client system with said compliance mechanism upon discovery of a kernel streaming mechanism operable on said client computer; and

governing said media content from said kernel-mode media device driver to a media device driver coupled with said compliance mechanism via said data path, for selectively restricting output of said media content.

104. A system of preventing unauthorized recording of electronic media comprising:

means for activating a compliance mechanism to control a data path of a client system, said activating in response to said client system receiving media content, said data path a path of a kernel-mode media device driver in an operating system operable on said client system, said compliance mechanism coupled to said client system and operable in conjunction with a media content presentation application coupled to said client system and operable thereon; and

means for directing said media content from said kernel-mode media device driver to a media device driver via said data path controlled by said compliance mechanism, for selectively restricting output of said media content.

105. The method according to Claim 102, the computer readable medium according to Claim 103, or the system according to Claim 104, wherein said media content is from a source coupled with said client system, wherein said source is from the group consisting of:

a network, a personal communication device, a satellite radio feed, a cable television radio input, a set-top box, an electronic media device, a media storage device, a media storage device inserted in a media device player, a media player application, and a media recorder application.

106. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:

preventing said media content from being returned from said kernel-mode media device driver to a recording application coupled to said client system from recording said media content when said recording violates a usage restriction applicable to said media content.

107. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:

allowing said media content to be returned from said kernel-mode device driver to a recording application coupled to said client system to record said media

content when said recording complies with a usage restriction applicable to said media content.

108. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:
restricting said client system to have said media device driver implemented as a default media device driver.

109. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:
authorizing said client system to receive said media content.

110. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:
accessing an indicator associated with said media content for indicating to said compliance mechanism a usage restriction applicable to said media content.

111. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:
accessing an indicator corresponding to said media content for indicating to said compliance mechanism a usage restriction applicable to said media content.

112. The method according to Claim 102 or the computer readable medium according to Claim 103, wherein said kernel-mode media device driver is part of an operating system operable on said client system.

113. The method according to Claim 102 or the method of the computer readable medium according to Claim 103, wherein said method further comprises:
altering said compliance mechanism in response to a change in said usage restriction, said usage restriction comprising a copyright restriction or licensing agreement applicable to said media content.

114. The method according to Claim 102 or the computer readable medium according to Claim 103 wherein said media device driver is selected as a default media device.

115. The system according to Claim 104 further comprising:
means for allowing said media content to be returned from said kernel-based media device driver to a recording application operable on said client system to allow recording of said media content when said recording complies with a usage restriction applicable to said media content.

116. The system according to Claim 104 further comprising:
means for preventing said media content from being returned from said kernel-based media device driver to a recording application operable on said client system to prohibit recording of said media content when said recording violates a usage restriction applicable to said media content.

117. The system according to Claim 104 further comprising:
means for restricting said client system to have said media device driver selected as a default media device.

118. The system according to Claim 104 further comprising:
means for authorizing said client system to receive said media content.

119. The system according to Claim 104 further comprising:
means for accessing an indicator for indicating to said compliance mechanism said usage restriction applicable to said media content, said indicator attached to said media content.

120. The system according to Claim 104 wherein said means for directing is activated upon detection of a kernel streaming mechanism operable on said client system.

121. The system according to Claim 104 further comprising:

means for altering said compliance mechanism in response to changes in said usage restriction, said usage restriction a copyright restriction or licensing agreement applicable to said media content.

122. A method for preventing unauthorized access to protected media disposed on a media storage device, said method comprising:

activating an autorun mechanism disposed on said media storage device in response to a device drive coupled with a computer system receiving said media storage device, said autorun mechanism for initiating installing a compliance mechanism on said computer system;

installing said compliance mechanism on said computer system, said compliance mechanism communicatively coupled with said computer system when installed thereon, said compliance mechanism for enforcing compliance with a usage restriction applicable to said protected media;

obtaining control of a data input pathway operable on said computer system; and

preventing said protected media on said media storage device from being captured by an extractor mechanism via said data input pathway while enabling presentation of said protected media.

123. A system for preventing unauthorized access to protected media on a media storage device, said system comprising:

a compliance mechanism disposed on said media storage device, said compliance mechanism configured to be installed on and communicatively coupled with a computer system, said compliance mechanism for complying with a usage restriction applicable to said protected media;

an autorun mechanism disposed on said media storage device, said autorun mechanism configured to install said compliance mechanism and a presentation mechanism on said computer system; and

wherein said compliance mechanism is configured to prevent capturing of said protected media by an extraction mechanism via a data input pathway on said computer system while presenting said protected media.

124. A computer readable medium for storing computer implementable instructions, said instructions for causing a computer system to perform a method of preventing access to media on a media storage device, said method comprising:

invoking an autorun protocol disposed on said media storage device in response to a device drive coupled with said computer system receiving said media storage device, said autorun protocol for installing a compliance mechanism on said computer system;

disposing said compliance mechanism on said computer system, said compliance mechanism communicatively coupled with said computer system when disposed thereon, said compliance mechanism for providing compliance with a usage restriction applicable to said protected media;

obtaining control of a data input pathway of said computer system with a filter driver coupled with said compliance mechanism and said computer system, said filter driver installed during said disposing of said compliance mechanism; and

preventing said protected media on said media storage device from being captured by an extraction mechanism via said data input pathway while enabling presentation of said protected media.

125. The method according to Claim 122, the system according to Claim 123, or the computer readable medium according to Claim 124, wherein said usage restriction is a copyright restriction or a licensing agreement applicable to said protected media.

126. The method according to Claim 122, the system according to Claim 123, or the computer readable medium according to Claim 124, wherein said media storage device upon which said protected media is disposed is from a group of media storage devices, said group consisting of:

a compact disk (CD), a mini CD, a digital versatile disk (DVD), a mini DVD, a compact flash card, a secure digital (SD) card, a memory stick, a digital audio tape (DAT), a digital video tape (DVT), a holographic storage object, a magneto-optical disk, a multi-layer fluorescent disk, an optical disk, and a magnetic disk.

127. The method according to Claim 122 further comprising:

bypassing said installing said compliance mechanism on said computer system if an instance of said compliance mechanism is predisposed on said computer system.

128. The method according to Claim 122 further comprising:
initiating a communication session between said computer system and a network to which said computer system is coupled and from which said compliance mechanism is available.

129. The method according to Claim 128 further comprising:
comparing said compliance mechanism present on said computer system with said compliance mechanism available from said network.

130. The method according to Claim 129 further comprising:
updating said compliance mechanism on said computer system accordingly.

131. The method according to Claim 130 further comprising:
invoking a presentation mechanism coupled with said computer system, said presentation mechanism for presenting said protected media so a user of said computer system can experience said protected media, said presentation mechanism authorized in accordance with said compliance mechanism to present said protected media.

132. The method according to Claim 130 further comprising:
installing a presentation mechanism on said computer system to enable said computer system to present said protected media so a user of said computer system can experience said protected media, said presentation mechanism authorized in accordance with said compliance mechanism to present said protected media.

133. The method according to Claim 122 wherein said autorun mechanism is activated in response to detection of a usage restriction indicator disposed on said media storage device, subsequent to said device drive receiving said media storage device.

134. The method according to Claim 122 wherein said autorun mechanism is activated in response to detection of a selection of an icon representing said protected media, said icon visibly displayed to a user of said computer system via a display device coupled with said computer system.

135. The method according to Claim 122 wherein a filter driver coupled to and operable in conjunction with said compliance mechanism controls said data input pathway, said filter driver disposed on said media storage device.

136. The method according to Claim 122 or the method of the computer readable medium according to Claim 124, wherein said method further comprises:
deactivating said compliance mechanism upon detection of removal of said media storage device from said computer system.

137. The method according to Claim 122 or the method of the computer readable medium according to Claim 124, wherein said method further comprises:
uninstalling said compliance mechanism upon detection of removal of said media storage device from said computer system.

138. The system according to Claim 123 further comprising a presentation mechanism disposed on said media storage device, said presentation mechanism configured to present said protected media in accordance with said compliance mechanism, said presentation mechanism configured to be installed on said computer system.

139. The system according to Claim 123 wherein said compliance mechanism further comprises a filter driver configured to be coupled with said compliance mechanism and said data input pathway, said filter driver for controlling said data input pathway.

140. The system according to Claim 123 wherein said compliance mechanism is configured to initiate a communication session between said computer

system and a network to which said computer system is coupled and from which a compliance mechanism is available.

141. The system according to Claim 140 wherein said compliance mechanism is configured to compare said compliance mechanism on said computer system with said compliance mechanism available from said network and to update said compliance mechanism on said computer system accordingly.

142. The system according to Claim 123 wherein said autorun protocol is configured to initiate installation of said compliance mechanism in response to said media storage device being inserted in a device drive coupled with computer system.

143. The system according to Claim 123 wherein said autorun protocol is configured to initiate installation of said compliance mechanism in response to a detection of a usage restriction indicator disposed on said media storage device, subsequent to a device drive coupled with said computer system receiving said media storage device.

144. The system according to Claim 123 wherein said autorun protocol is configured to initiate installation of said compliance mechanism in response to detection of a selection of an icon representing said protected media, said icon visibly displayed to a user of said computer system via a display device coupled with said computer system.

145. The system according to Claim 123 wherein said autorun protocol is configured to bypass said install upon detection of a compliance mechanism present on said computer system.

146. The system according to Claim 123 wherein said compliance mechanism is configured to be deactivated upon detection of removal of said media storage device from said computer system.

147. The system according to Claim 123 wherein said compliance mechanism is configured to be uninstalled upon detection of removal of said media storage device from said computer system.

148. The computer readable medium according to Claim 124 wherein said method further comprises:

bypassing said disposing said compliance mechanism on said computer system if a compliance mechanism is predisposed thereon.

149. The computer readable medium according to Claim 124, wherein said method further comprises:

commencing a communication session between said computer system and a network to which said computer system is coupled and from which said compliance mechanism is available.

150. The computer readable medium according to Claim 149 wherein said method further comprises:

comparing said compliance mechanism on said computer system with said compliance mechanism available from said network and updating said compliance mechanism on said computer system accordingly.

151. The computer readable medium according to Claim 150, wherein said method further comprises:

activating a presentation mechanism coupled with said computer system, said presentation mechanism for presenting said protected media so a user of said computer system can experience said protected media, said presentation mechanism authorized to present said protected media in accordance with said compliance mechanism.

152. The computer readable medium according to Claim 150, wherein said method further comprises:

installing a presentation mechanism on said computer system to enable said computer system to present said protected media so a user of said computer system

can experience said protected media, said presentation mechanism authorized to present said protected media in accordance with said compliance mechanism.

153. The computer readable medium according to Claim 124, wherein said autorun protocol is invoked in response to detection of a usage restriction indicator disposed on said media storage device, subsequent to said device drive receiving said media storage device.

154. The computer readable medium according to Claim 124, wherein said autorun protocol is invoked in response to detection of a selection of an icon representing said protected media, said icon visibly displayed to a user of said computer system via a display device coupled with said computer system.

155. A method for controlling presentation of content on a media storage device, said method comprising:

verifying presence of a content presentation mechanism and a usage compliance mechanism on a computer system operated by a recipient to whom said media storage device is distributed, said usage compliance mechanism including a file system filter driver for controlling data reads associated with said content;

permitting said recipient to experience said content via said computer system provided said usage compliance mechanism is present on said computer system and said computer system is communicatively coupled with a network and wherein a server in said network authorizes said recipient to experience said content; and

presenting said content to said recipient via said content presentation mechanism, said content presentation mechanism communicatively coupled with said usage compliance mechanism, said content presentation mechanism enabled to present said content provided said content presentation mechanism is communicatively coupled with said server.

156. A system for controlling presentation of content on a media storage device comprising:

a detecting means for detecting presence of a usage compliance mechanism operable on a computer system operated by a recipient to whom said media storage

device is distributed, said detecting means also for detecting presence of a content presentation mechanism operable on said computer system, said usage compliance mechanism including a file system filter driver for controlling data reads associated with said content;

an authorizing means for authorizing said recipient to experience said content provided said usage compliance mechanism is installed on said computer system and said computer system is communicatively coupled with a network and wherein a server in said network issues authorization allowing said recipient to experience said content; and

a content presenting means for presenting said content to said recipient, said content presentation means communicatively coupled with said usage compliance mechanism, said content presenting means enabled to present said content to said recipient provided said content presentation means is communicatively coupled with said server.

157. A computer readable medium for storing computer implementable instructions, said instructions for causing a computer system to perform a method of controlling presentation of content on a media storage device, said method comprising:

determining presence of a usage compliance mechanism and a content presentation mechanism on a computer system operated by a recipient to whom said media storage device is distributed, said usage compliance mechanism including a file system filter driver for controlling data reads associated with said content;

allowing said recipient to experience said content via said computer system provided said usage compliance mechanism is present on said computer system and said computer system is communicatively coupled with a network and wherein a server in said network authorizes said recipient to experience said content; and

presenting said content to said recipient via said content presentation mechanism, said content presentation mechanism communicatively coupled with said usage compliance mechanism, said content presentation mechanism enabled to present said content provided said content presentation mechanism is communicatively coupled with said server.

158. The method as recited in Claim 155 or the method of the computer readable medium of Claim 157, wherein said method further comprises:

installing said usage compliance mechanism on said computer system when said usage compliance mechanism is not present on said computer system and installing said content presentation mechanism on said computer system when said content presentation mechanism is not present on said computer system.

159. The method according to Claim 155 or the computer readable medium according to Claim 157m wherein an autorun mechanism disposed on said media storage device initiates installation of said usage compliance mechanism and said content presentation mechanism on said computer system in response to said computer system receiving said media storage device when said usage compliance mechanism and said content presentation mechanism are not present on said computer system.

160. The method as recited in Claim 155 or the method of the computer readable medium of Claim 157, wherein said method further comprises:

encrypting said content prior to disposal of said content on said media storage device.

161. The method or computer readable medium according to Claim 160, wherein said encrypting comprises a first encryption applied to said content, and wherein a second encryption is applied to said content and to said first encryption.

162. The method or computer readable medium according to Claim 160, wherein said encrypting comprises a unique first encryption applied to each instance of content when a plurality of content is disposed on said media storage device, and wherein a unique second encryption is applied to each said unique first encryption and associated content.

163. The method or computer readable medium according to Claim 160, wherein said encrypting further comprises a plurality of unique first encryptions

applied to each instance of content when a plurality of content is disposed on said media storage device, and wherein said second encryption is applied to each of said unique first encryptions and to said content.

164. The method or computer readable medium according to Claim 161, wherein said method further comprises:

decrypting said second encryption with said file system filter driver using a second decryption key stored on said server.

165. The method or computer readable medium according to Claim 161, wherein said method further comprises:

decrypting said first encryption with said content presentation mechanism using a first decryption key stored on said server, said decrypting said first encryption concurrent with presenting said content.

166. The method according to Claim 155 further comprising:

affixing a unique identifier on said media storage device.

167. The computer readable medium according to Claim 157 wherein said method further comprises:

depositing a unique identifier on said media storage device.

168. The method or computer readable medium according to Claim 167 wherein said unique identifier is a serial number, said serial number generated during disposition of said content on said media storage device.

169. The method or computer readable medium according to Claim 168, wherein said method further comprises:

watermarking said content via said content presentation mechanism during decryption of a first encryption applied to said content, said content presentation mechanism further causing said unique identifier to be watermarked onto an outgoing data stream containing said content.

170. The method as recited in Claim 155 or the method of the computer readable medium of Claim 157, wherein said method further comprises:

updating said content presentation mechanism and said usage compliance mechanism via said network.

171. The system according to Claim 156 further comprising:

an autorun means for automatically installing said usage compliance mechanism and for automatically installing said content presentation means when said usage compliance mechanism and said content presentation means are not detected on said computer system, said autorun means invoked in response to said computer system receiving said media storage device, said autorun means disposed on said media storage device.

172. The system according to Claim 156 further comprising:

an encrypting means for encrypting said content prior to disposition of said content on said media storage device, wherein said encrypting comprises a first encryption applied to said content and a second encryption applied to said content with said first encryption applied thereto.

173. The system according to Claim 172 further comprising:

a second decryption means for decrypting said second encryption, said decrypting of said second encryption performed by said file system filter driver and in conjunction with a second decryption key stored on said server.

174. The system according to Claim 172 further comprising:

a first decryption means for decrypting said content with said first encryption applied thereto, said decrypting of said first encryption performed by said content presenting means in conjunction with a first decryption key stored on said server, wherein said first encryption applied to said content is decrypted concurrent with said content presenting means presenting said content to said recipient.

175. The system according to Claim 156 further comprising:

an identification means for providing an unique identification of said media storage device, and wherein said unique identification is disposed on said media storage device.

176. The system according to Claim 175 further comprising:

a watermarking means for watermarking said content, said watermarking performed by said content presenting means and concurrent with presenting said content, said content presenting means further causing said unique identifier to be watermarked onto an outgoing data stream containing said content.

177. The system according to Claim 156 further comprising:

an updating means for updating said usage compliance mechanism and said content presenting means, said updating means communicatively coupled with said computer system and said network.

178. A method comprising:

transmitting via a communication network a first request for a media content list from a computer; and

transmitting to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request.

179. A computer readable medium having computer readable code embodied therein for causing a system to perform:

transmitting via a communication network a first request for a media content list from a computer; and

transmitting to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request.

180. A system comprising:

means for transferring via a communication network a first request for a media content list from a computer; and

means for transferring to said computer via said communication network said media content list together with a unique identification, in response to receiving said first request.

181. A system comprising:
a media content library;
a global media content delivery network; and
a security system.

182. A method comprising:
transmitting a first request for a media content list of a media content library from a client device; and
in response to receiving said first request, transmitting to said client device said media content list together with a unique identification.

183. A system comprising:
a media content library;
a global media content delivery network; and
an authorization system.

184. A method comprising:
detecting a media player application operable with a computer system, said media player application for enabling said computer system to present contents of a media file; and
governing within said media player application a function that enables non-compliance with a usage restriction applicable to said media file.

185. A computer readable medium for storing computer implementable instructions, said instructions for causing a compliance mechanism to perform a method of controlling interaction of a media file, said method comprising:
discovering a media player application operable within a client computer system, said media player application for presenting contents of a media file deliverable to said client computer system; and

regulating a function of said media player application that does not comply with usage restrictions applicable to said media file.

186. A system comprising:

means for detecting a media player application operable on a client computer system and for presenting contents of a media file; and

means for governing a function of said media player application that does not comply with a usage restriction applicable to a media file.

187. A method comprising:

activating a compliance mechanism in response to receiving media content by a client system; and

controlling a data output path of said client computer with said compliance mechanism.

188. A computer readable medium for storing computer implementable instructions, said instructions for causing a client system to perform a method of restricting recording of media content, said method comprising:

animating a compliance mechanism coupled to said client system; and

managing an output path of said client computer with said compliance mechanism.

189. A system comprising:

means for activating a compliance mechanism to control a data output path of a client system; and

means for directing media content to a custom media device via said data output path controlled by said compliance mechanism.

190. A method comprising:

activating a compliance mechanism in response to a client system receiving media content; and

controlling a data path of a kernel-mode media device driver of said client system with said compliance mechanism.

191. A computer readable medium for storing computer implementable instructions, said instructions for causing a client system to perform a method of restricting recording of media content, said method comprising:

animating a compliance mechanism coupled to said client system; and
managing a data path of a kernel-mode media device driver of said client system with said compliance mechanism.

192. A system comprising:

means for activating a compliance mechanism to control a data path of a client system; and

means for directing media content from a kernel-mode media device driver to a media device driver via said data path controlled by said compliance mechanism.

193. A method comprising:

activating an autorun mechanism disposed on a media storage device in response to a device drive coupled with a computer system receiving said media storage device, said autorun mechanism for initiating installing a compliance mechanism on said computer system; and

installing said compliance mechanism on said computer system.

194. A system comprising:

a compliance mechanism disposed on a media storage device; and
an autorun mechanism disposed on said media storage device.

195. A computer readable medium for storing computer implementable instructions, said instructions for causing a computer system to perform a method of preventing access to media on a media storage device, said method comprising:

invoking an autorun protocol disposed on said media storage device in response to a device drive coupled with said computer system receiving said media storage device, said autorun protocol for installing a compliance mechanism on said computer system; and

disposing said compliance mechanism on said computer system.

196. A method comprising:
verifying presence of a content presentation mechanism and a usage compliance mechanism on a computer system operated by a recipient; and
permitting said recipient to experience content via said computer system provided said usage compliance mechanism is present on said computer system.

197. A system comprising:
a detecting means for detecting presence of a usage compliance mechanism operable on a computer system operated by a recipient; and
an authorizing means for authorizing said recipient to experience content provided said usage compliance mechanism is installed on said computer system.

198. A computer readable medium for storing computer implementable instructions, said instructions for causing a computer system to perform a method of controlling presentation of content on a media storage device, said method comprising:
determining presence of a usage compliance mechanism and a content presentation mechanism on a computer system operated by a recipient; and
allowing said recipient to experience said content via said computer system provided said usage compliance mechanism is present on said computer system.

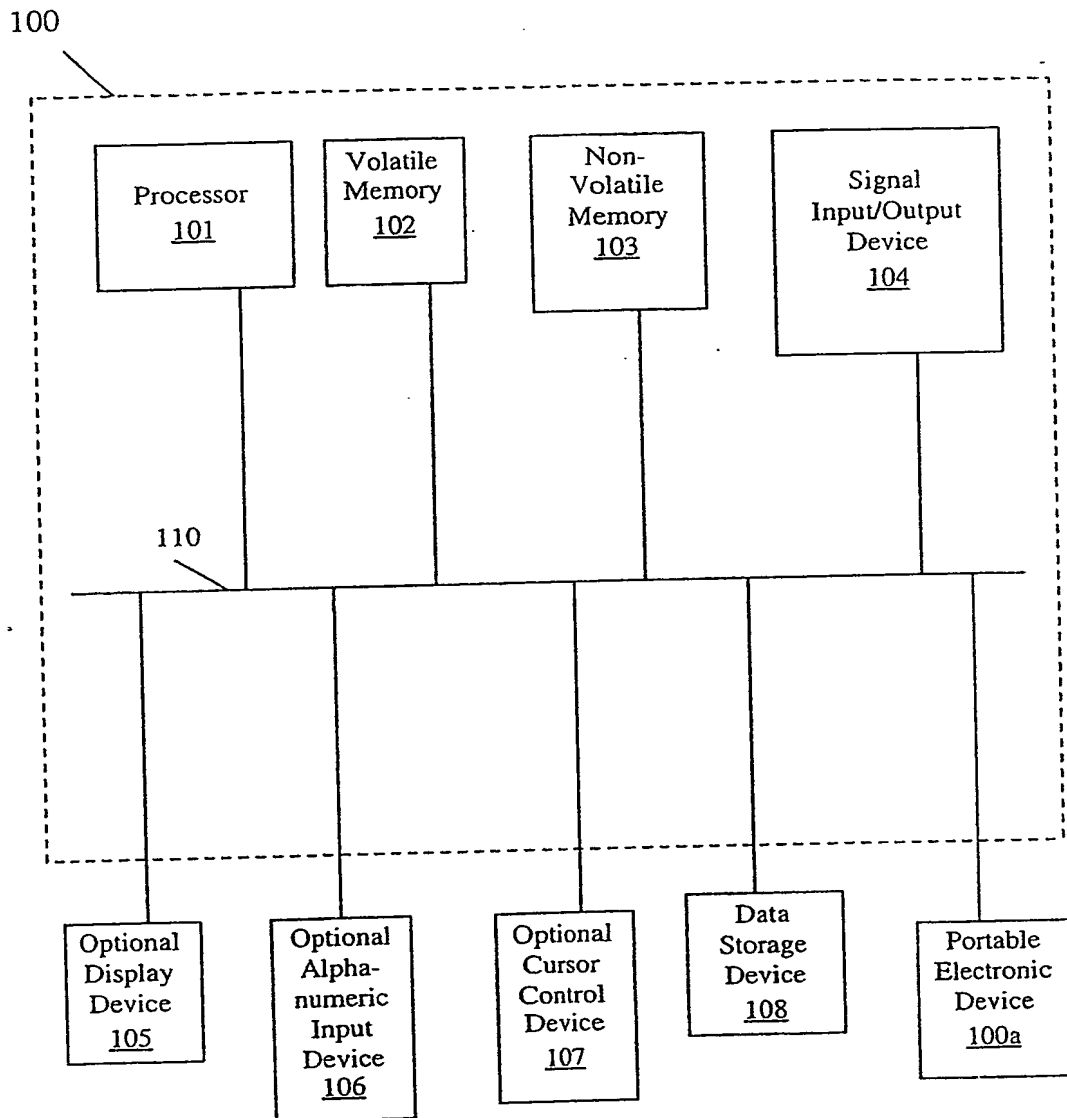


FIGURE 1

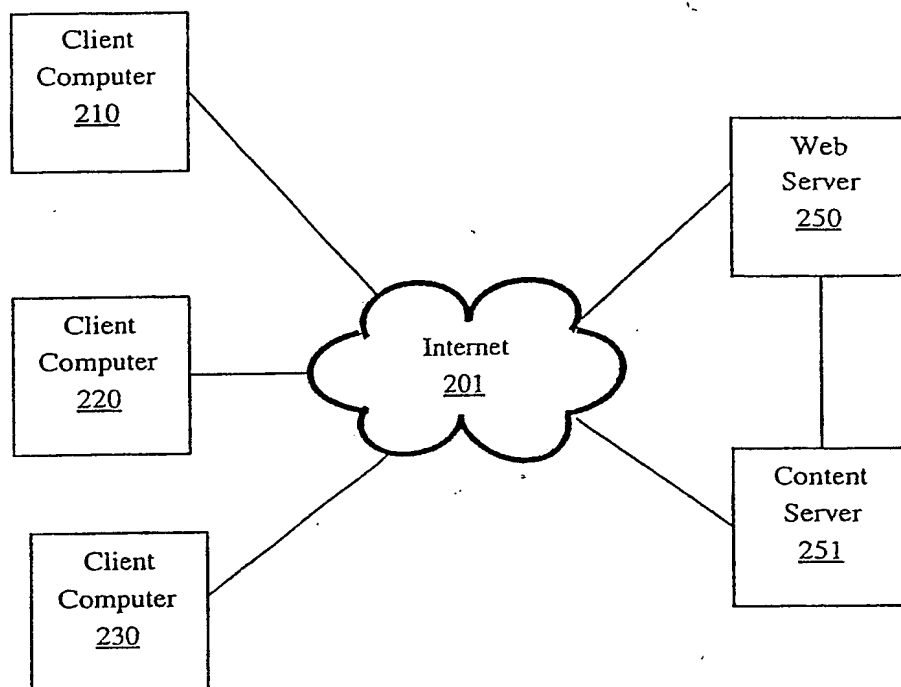
200

FIGURE 2

300

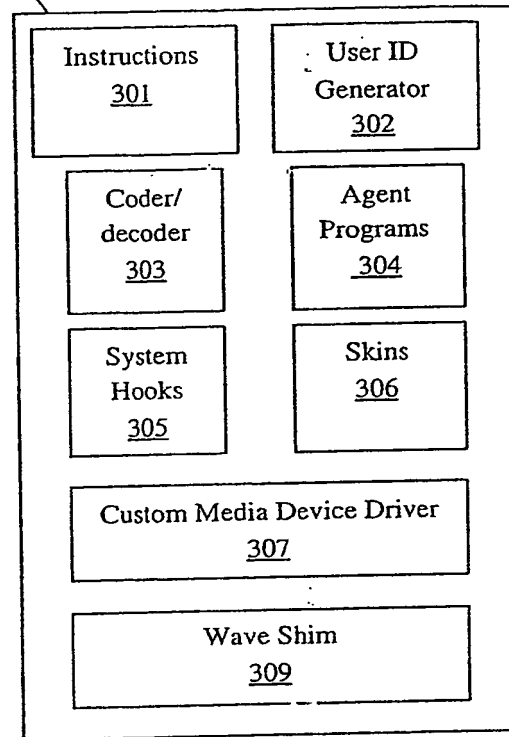


FIGURE 3

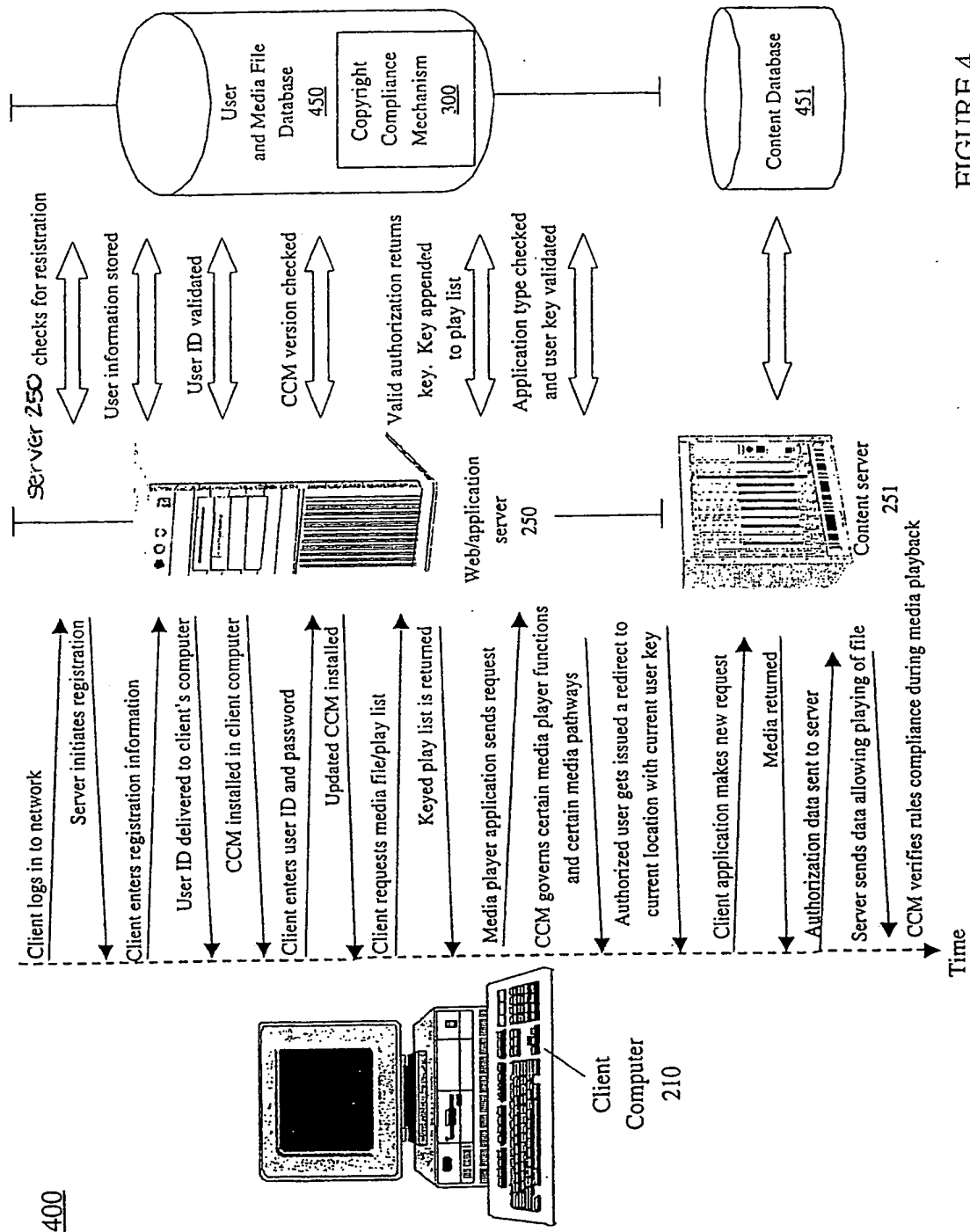


FIGURE 4

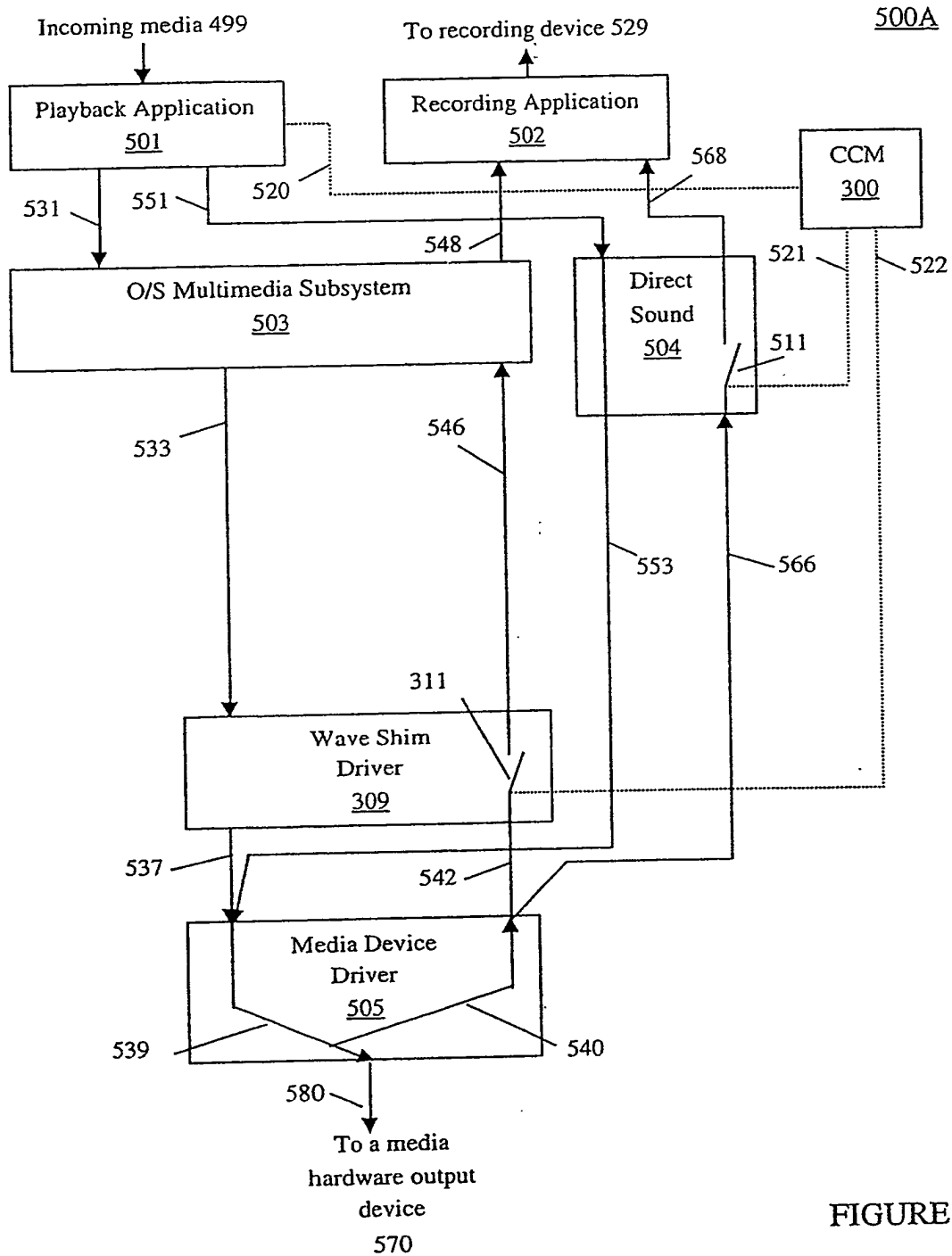


FIGURE 5A

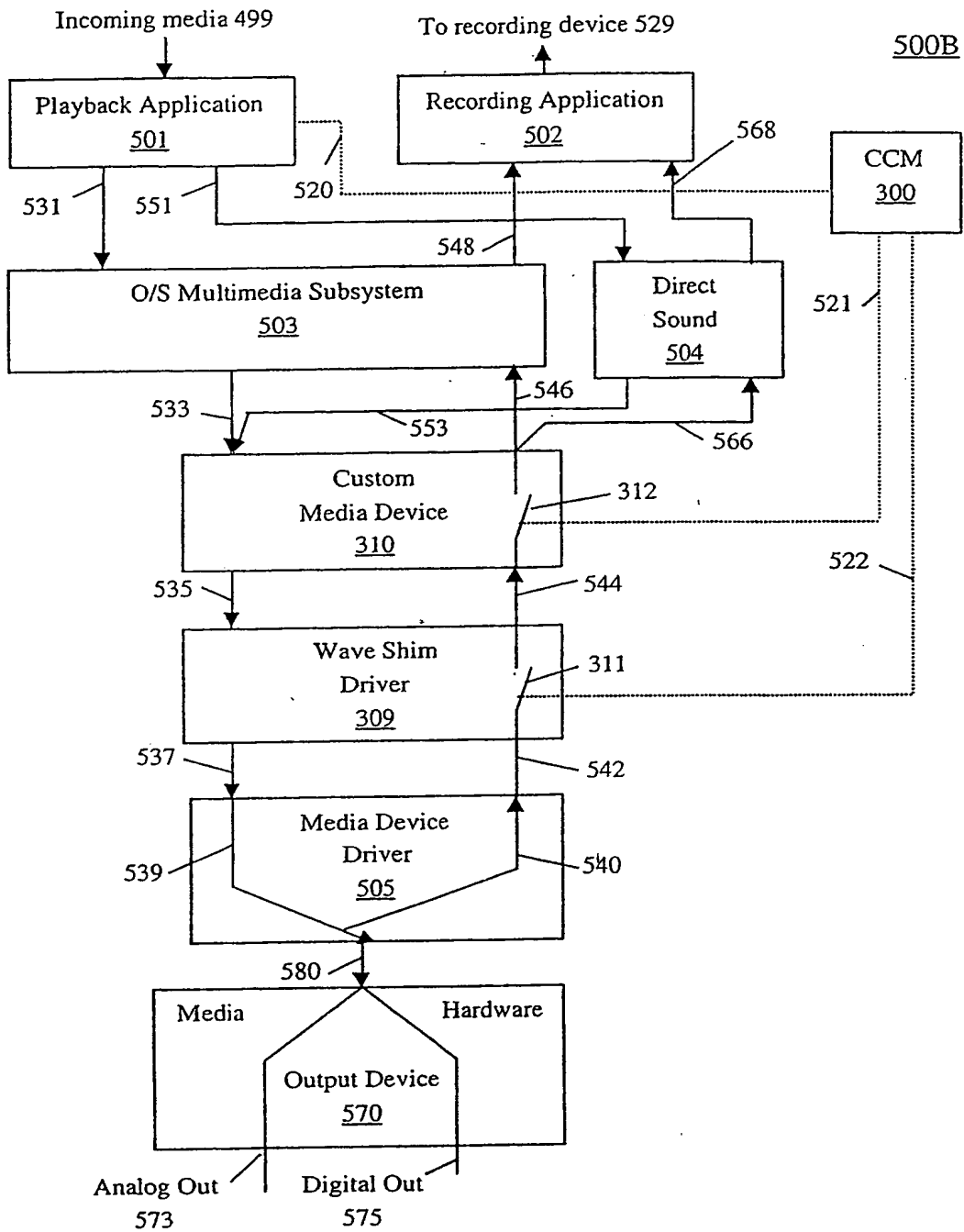
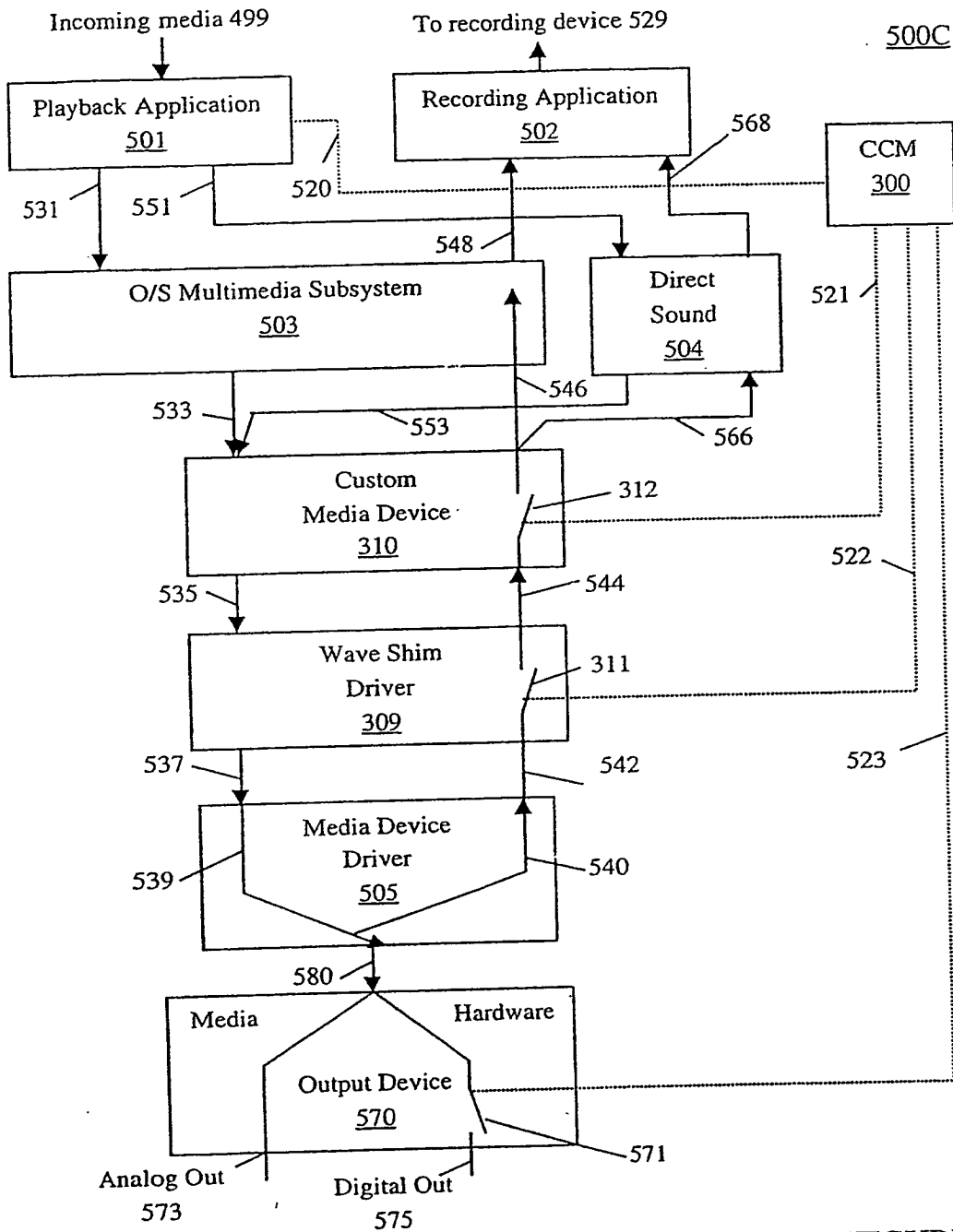


FIGURE 5B



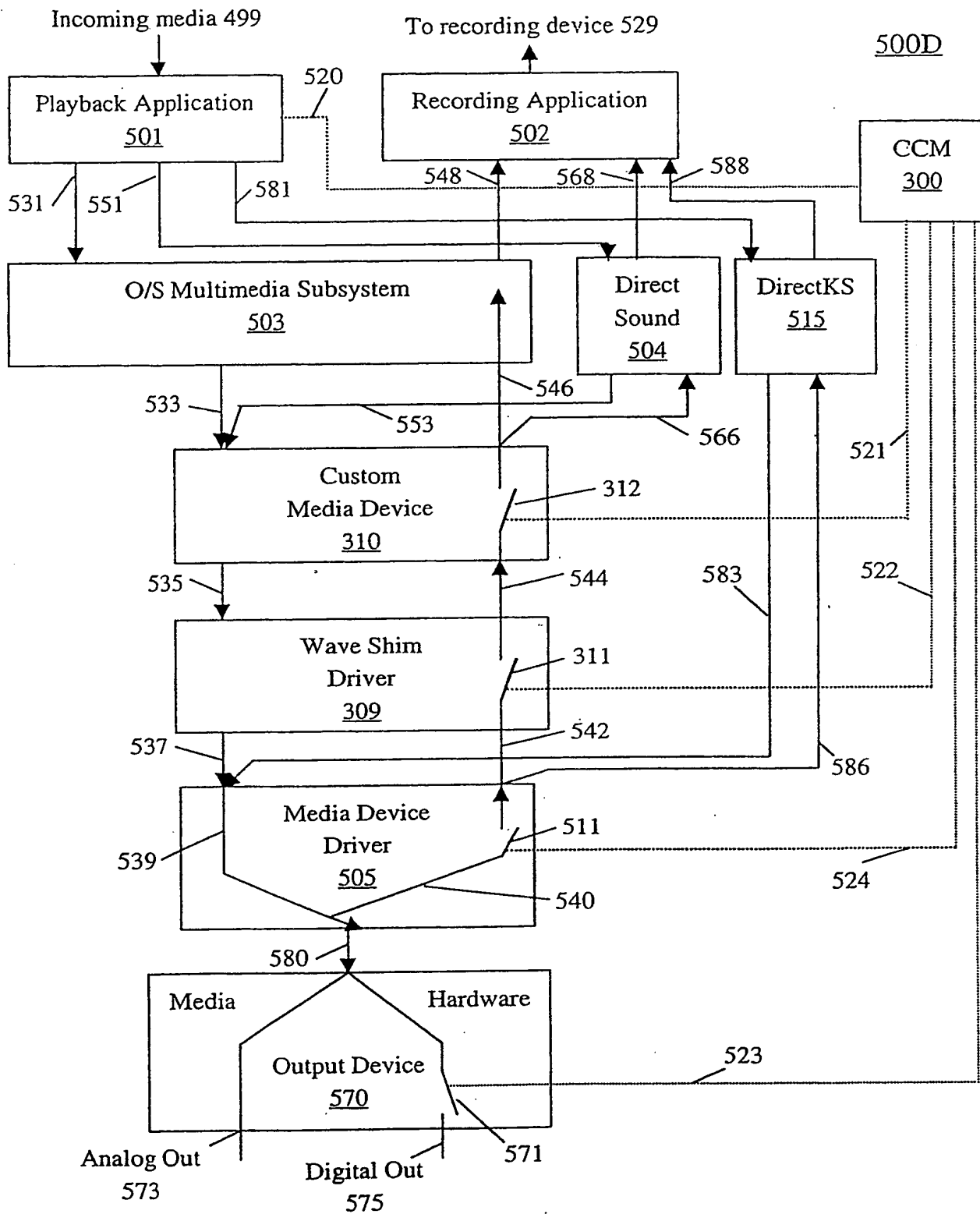


FIGURE 5D

600

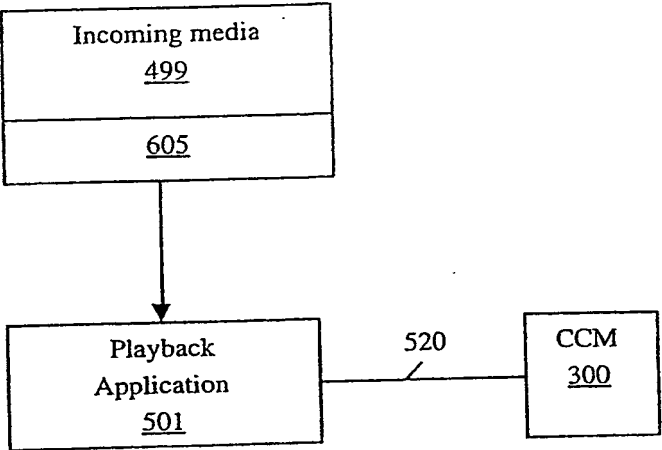


FIGURE 6A

700

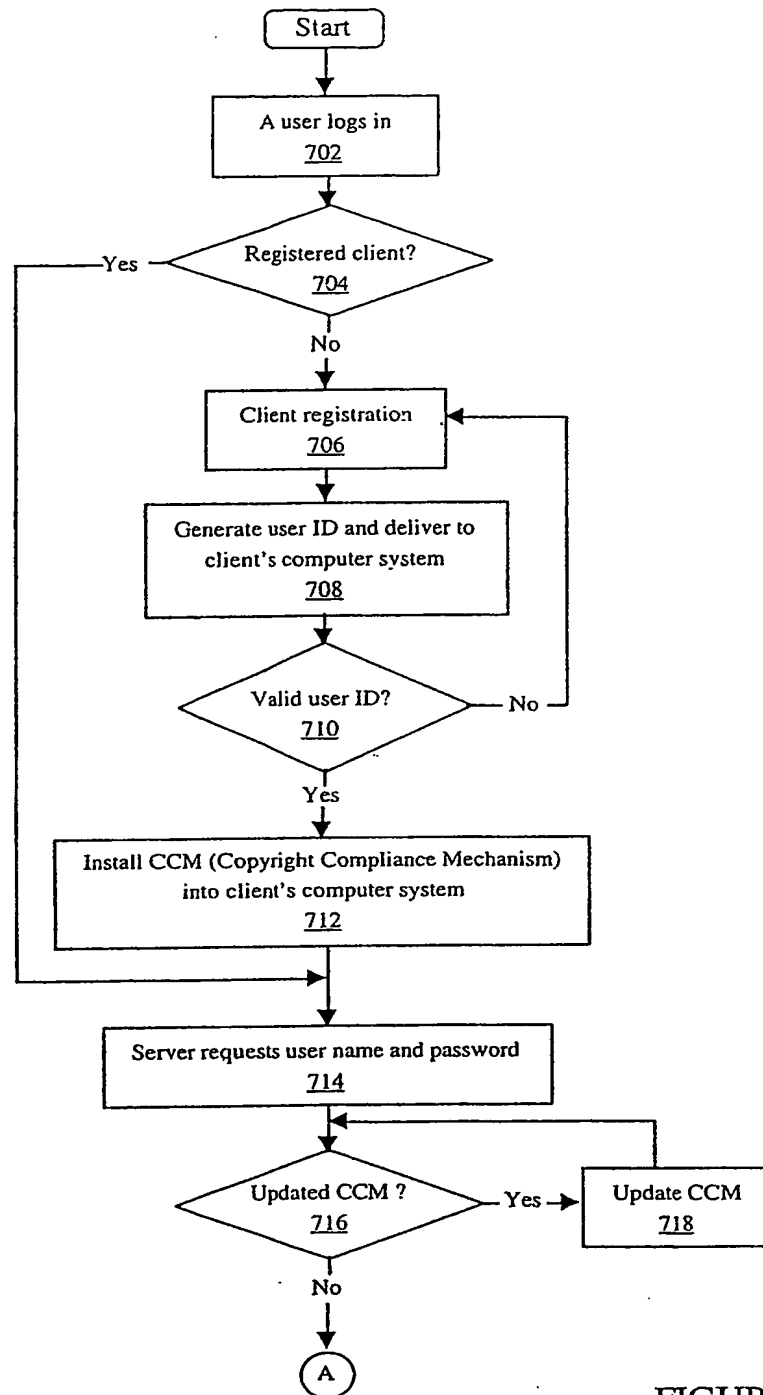


FIGURE 7A

700

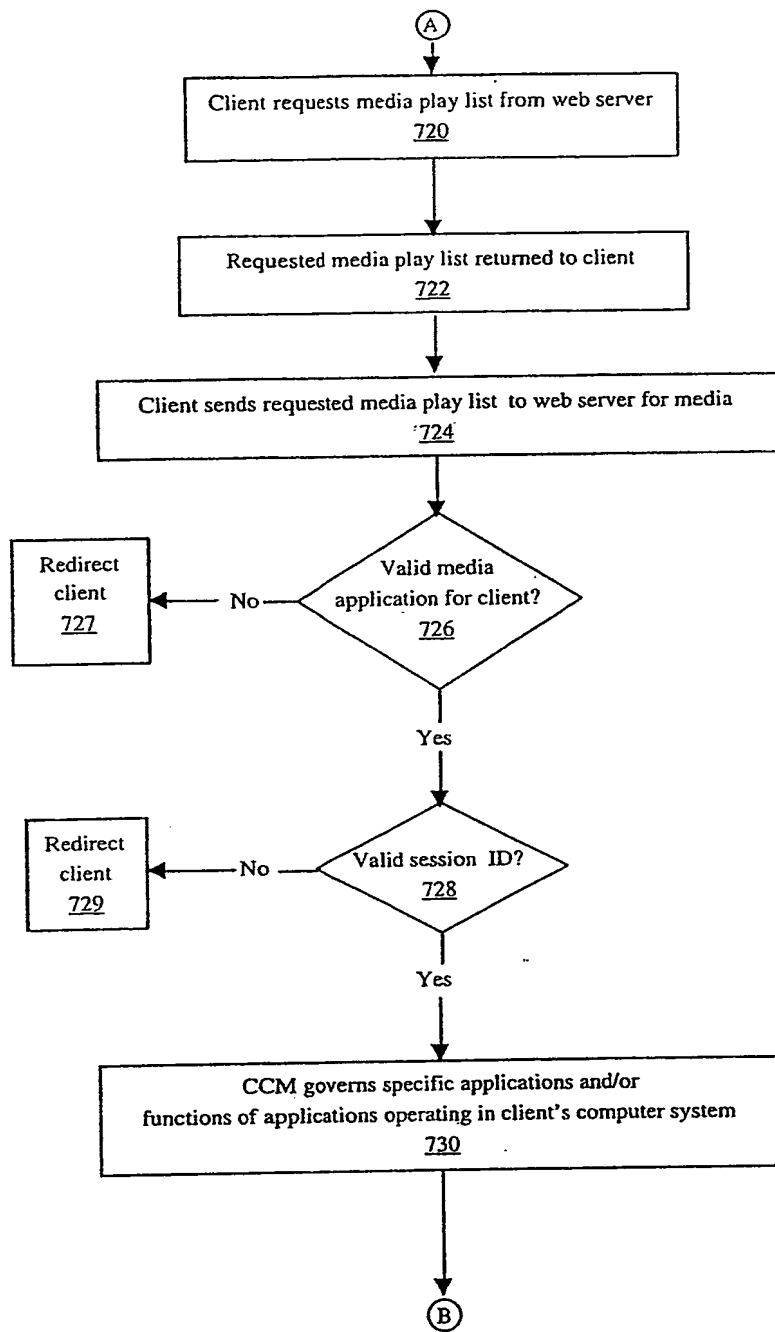


FIGURE 7B

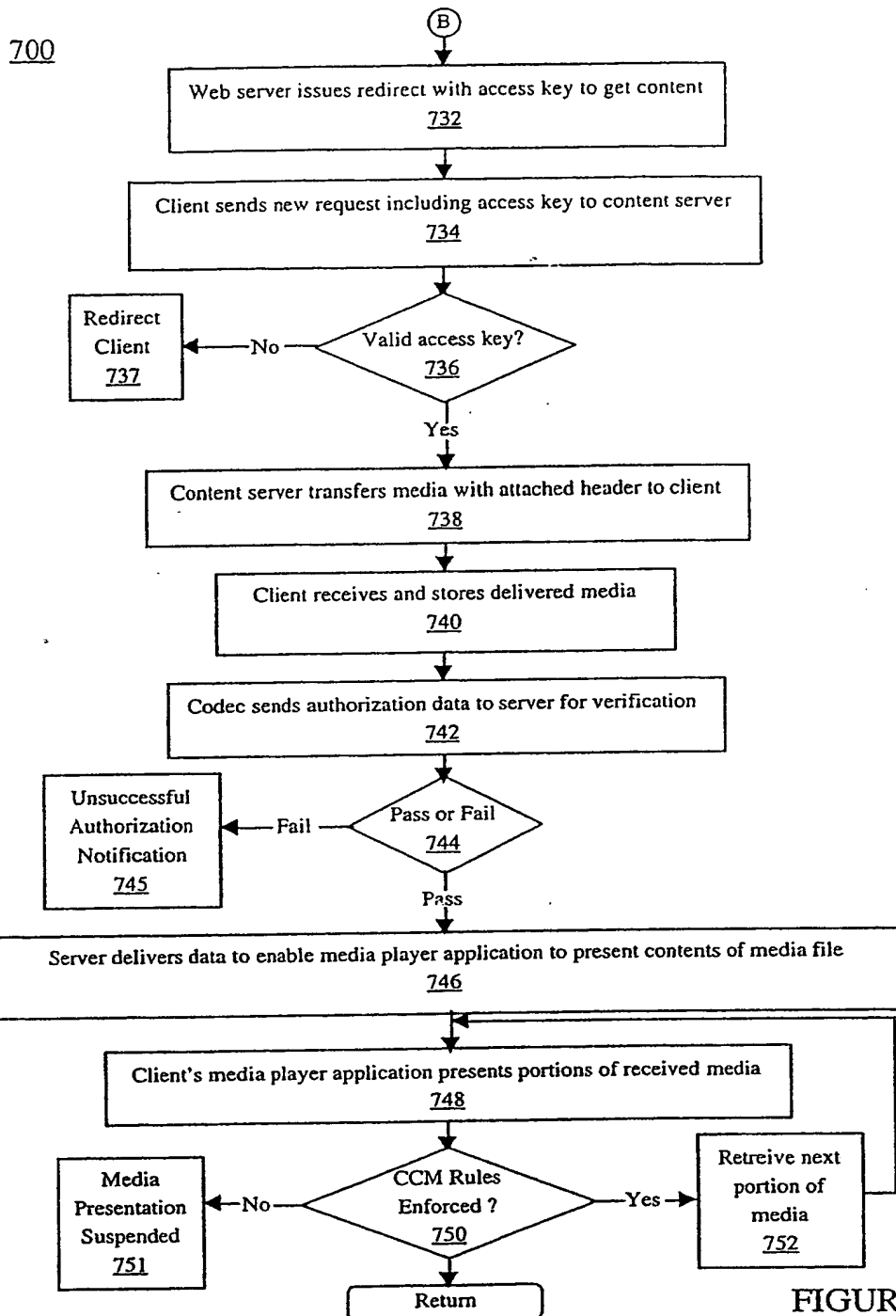


FIGURE 7C

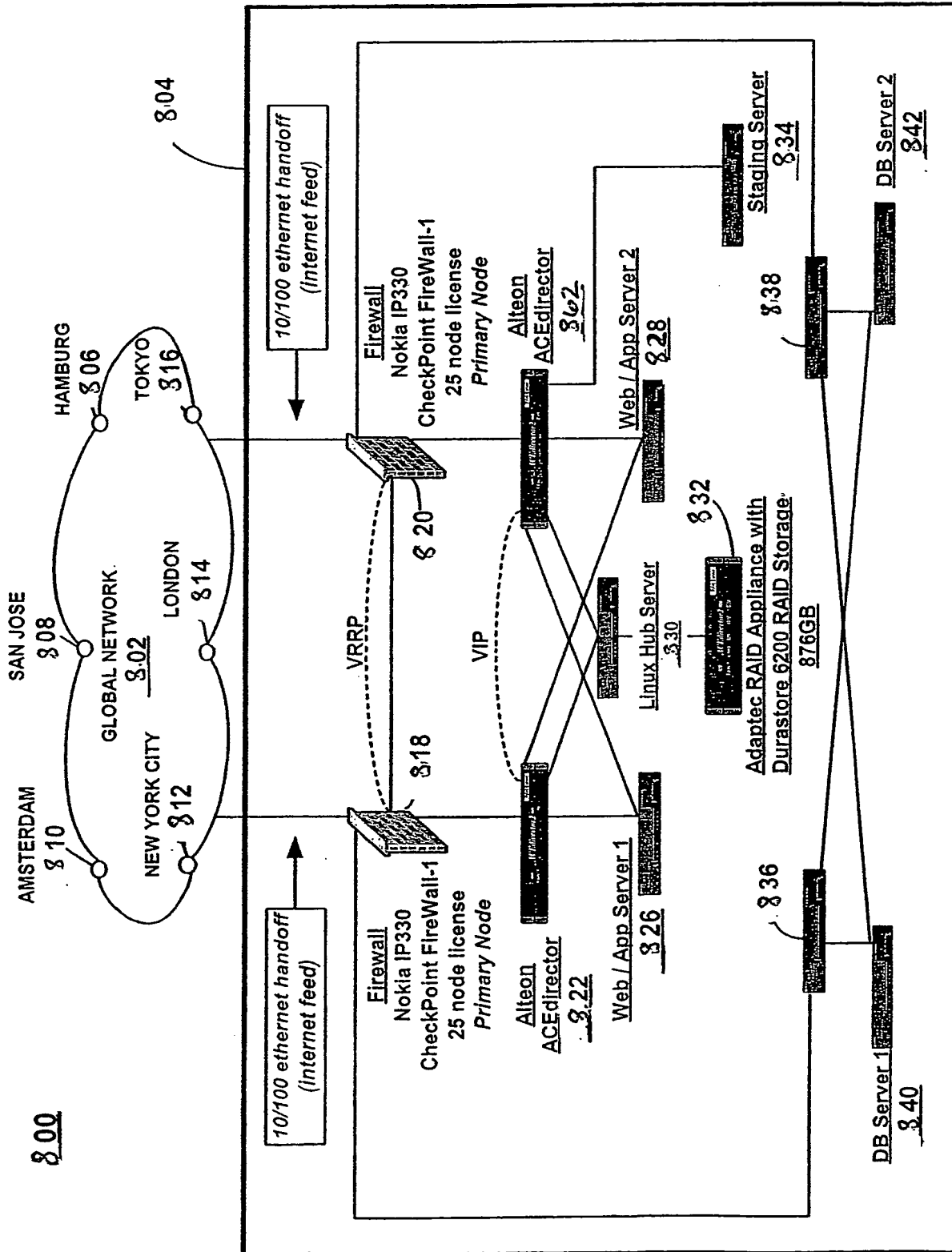


FIG. 8

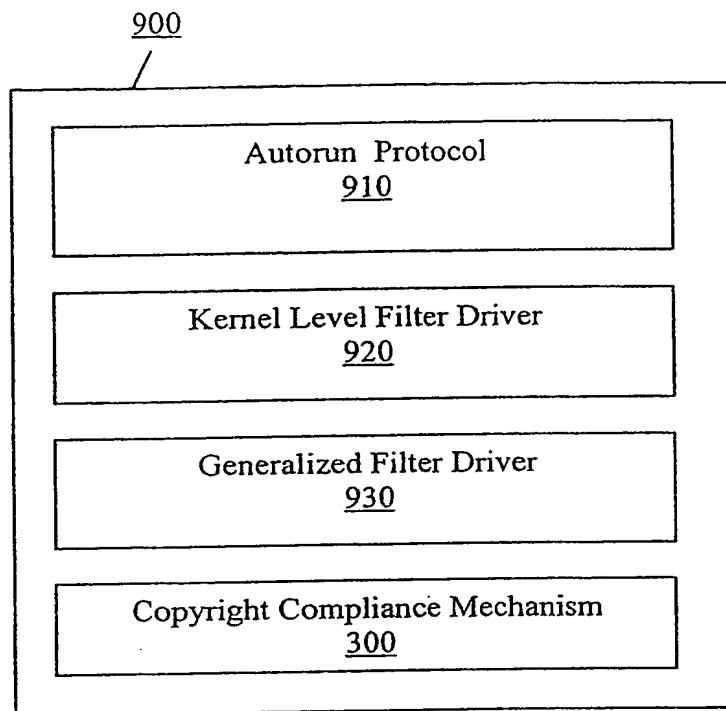


FIGURE 9

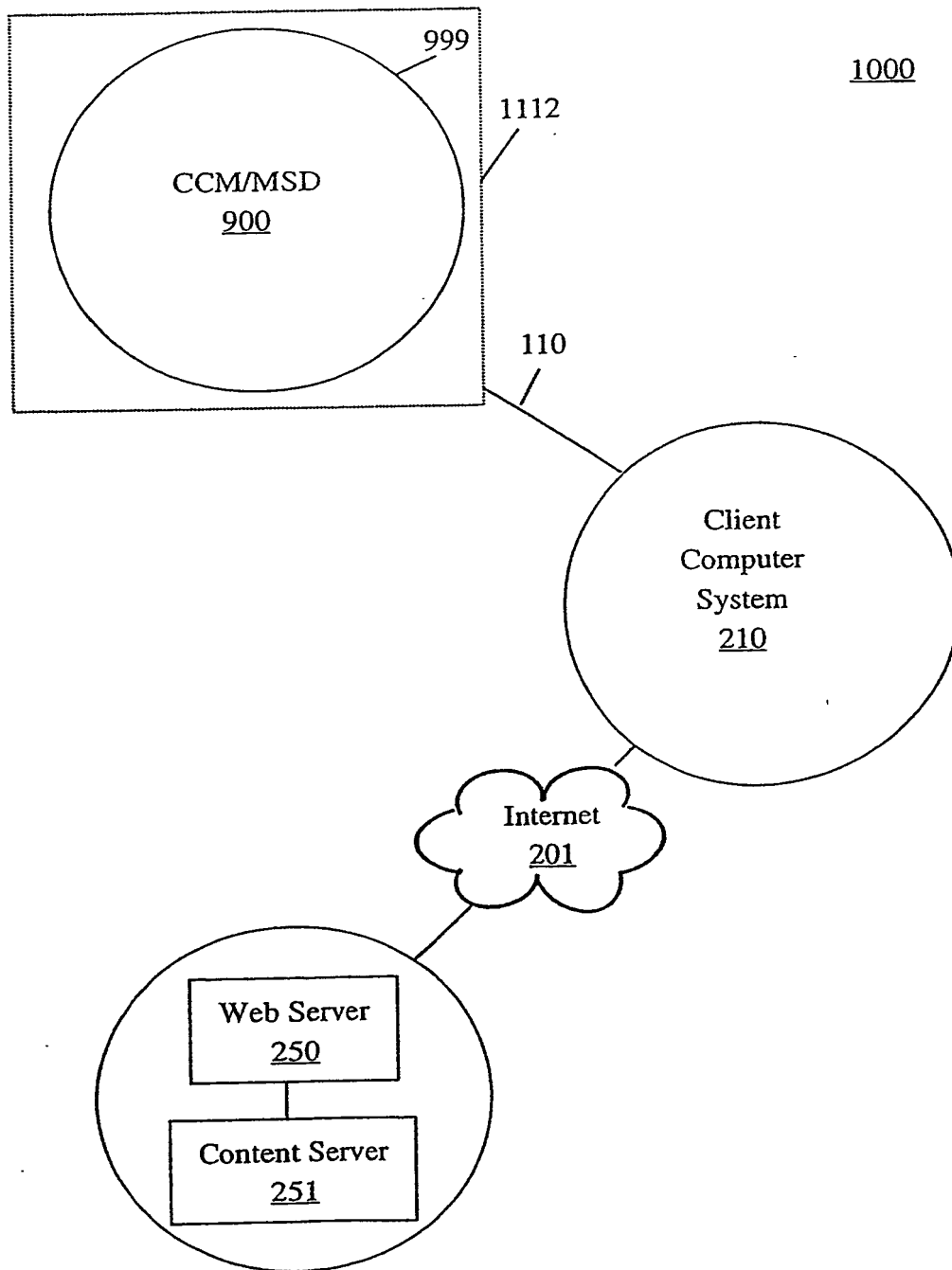


FIGURE 10

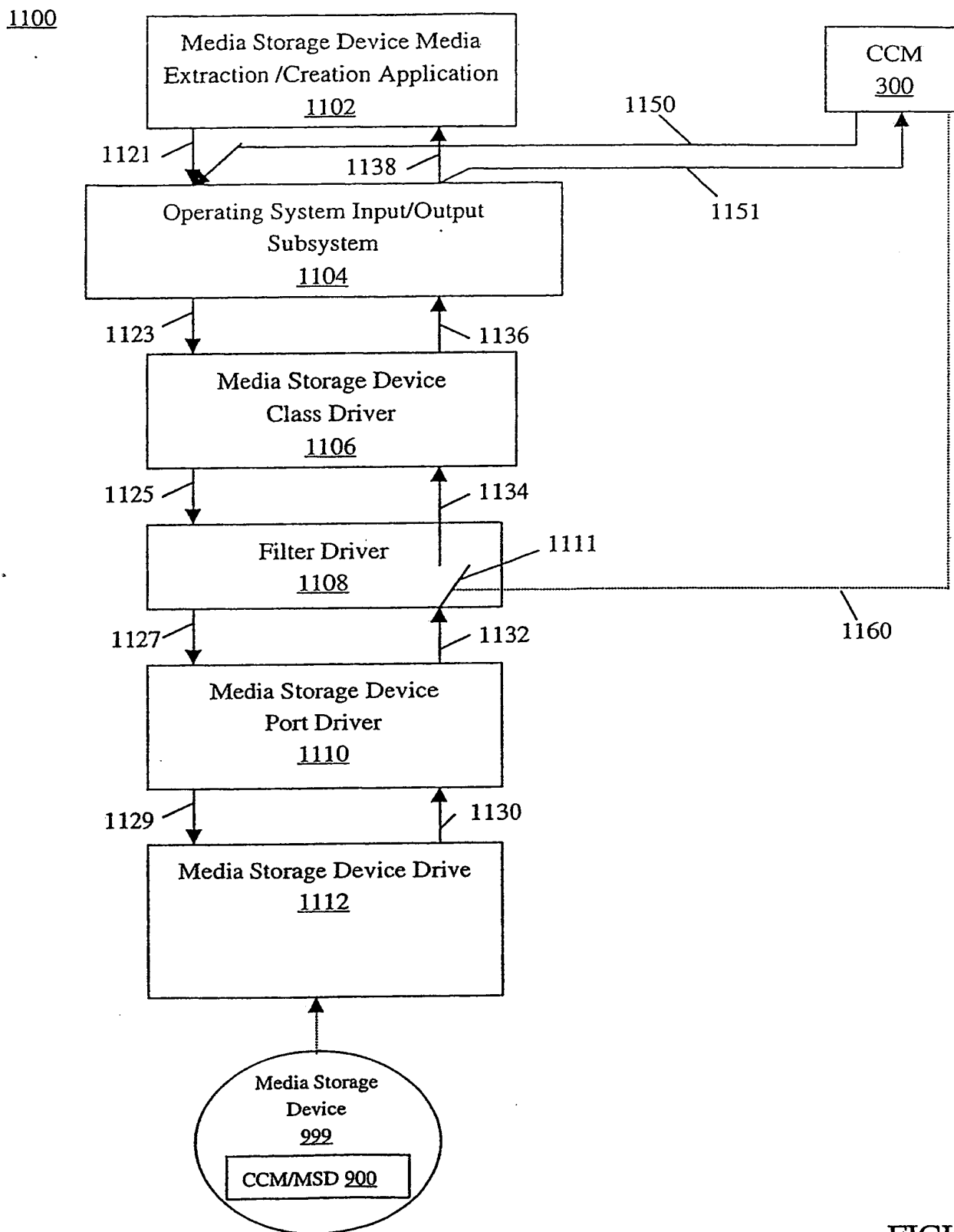


FIGURE 11

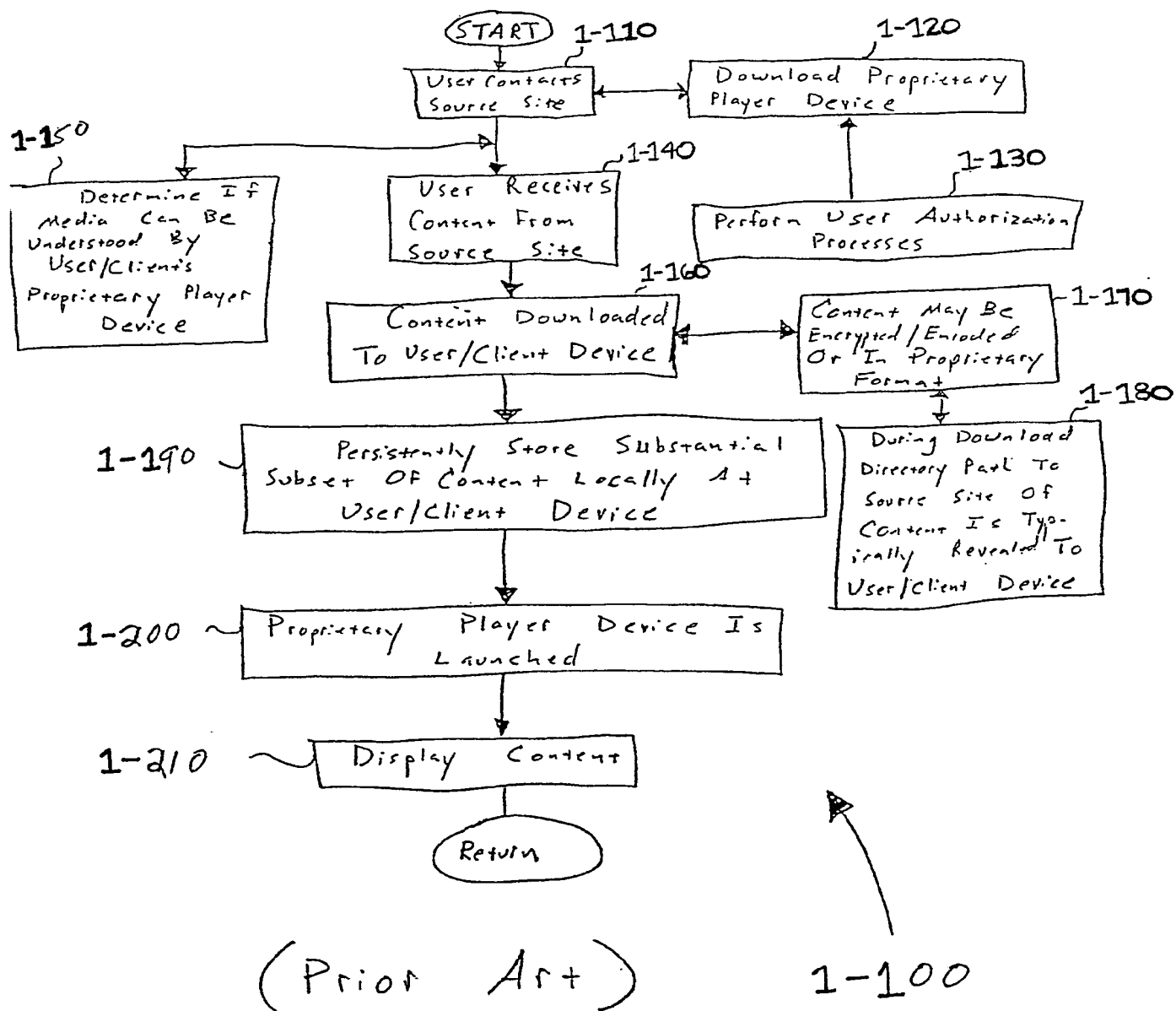


Fig. 1-1

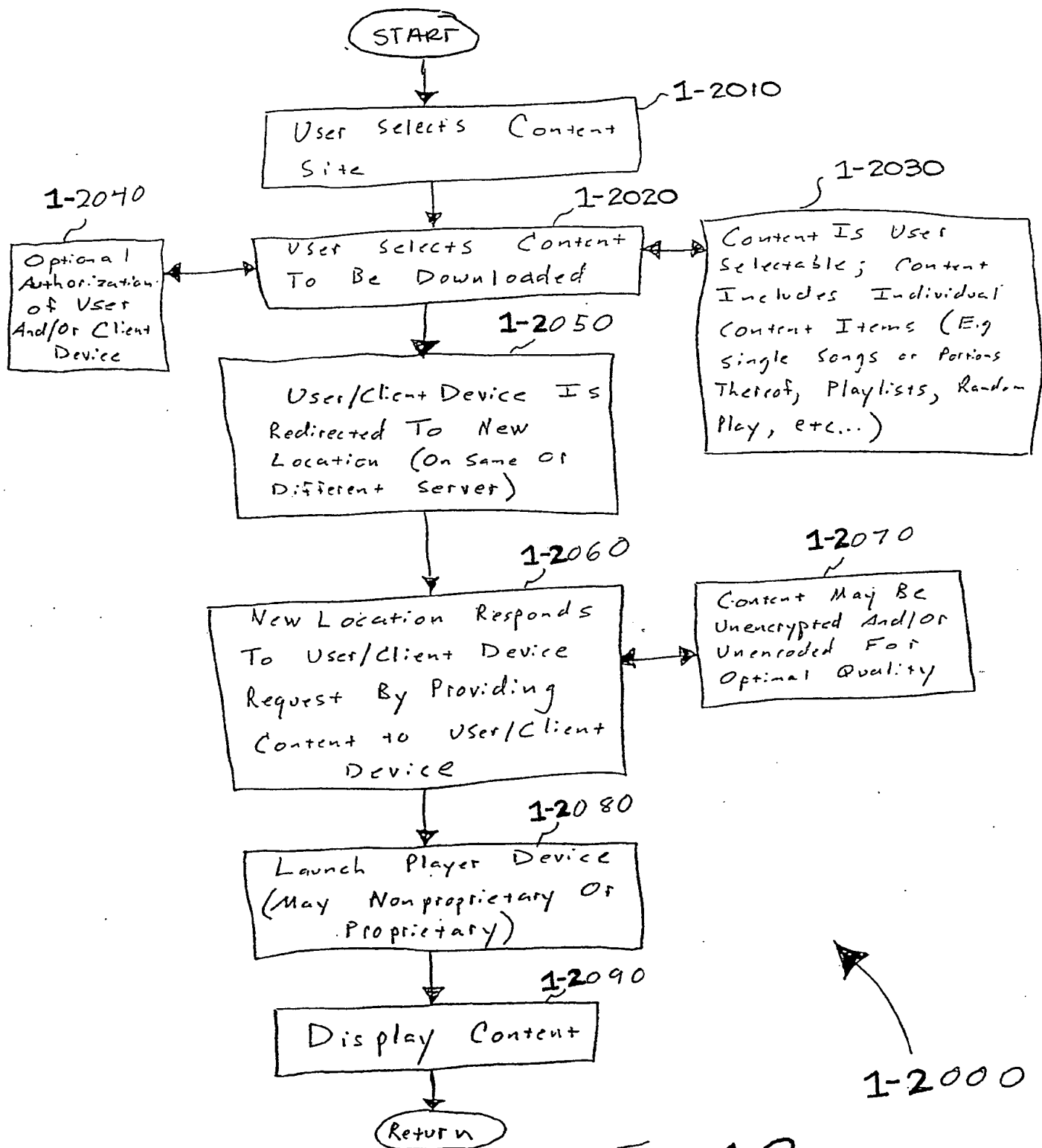


Fig. 1-2

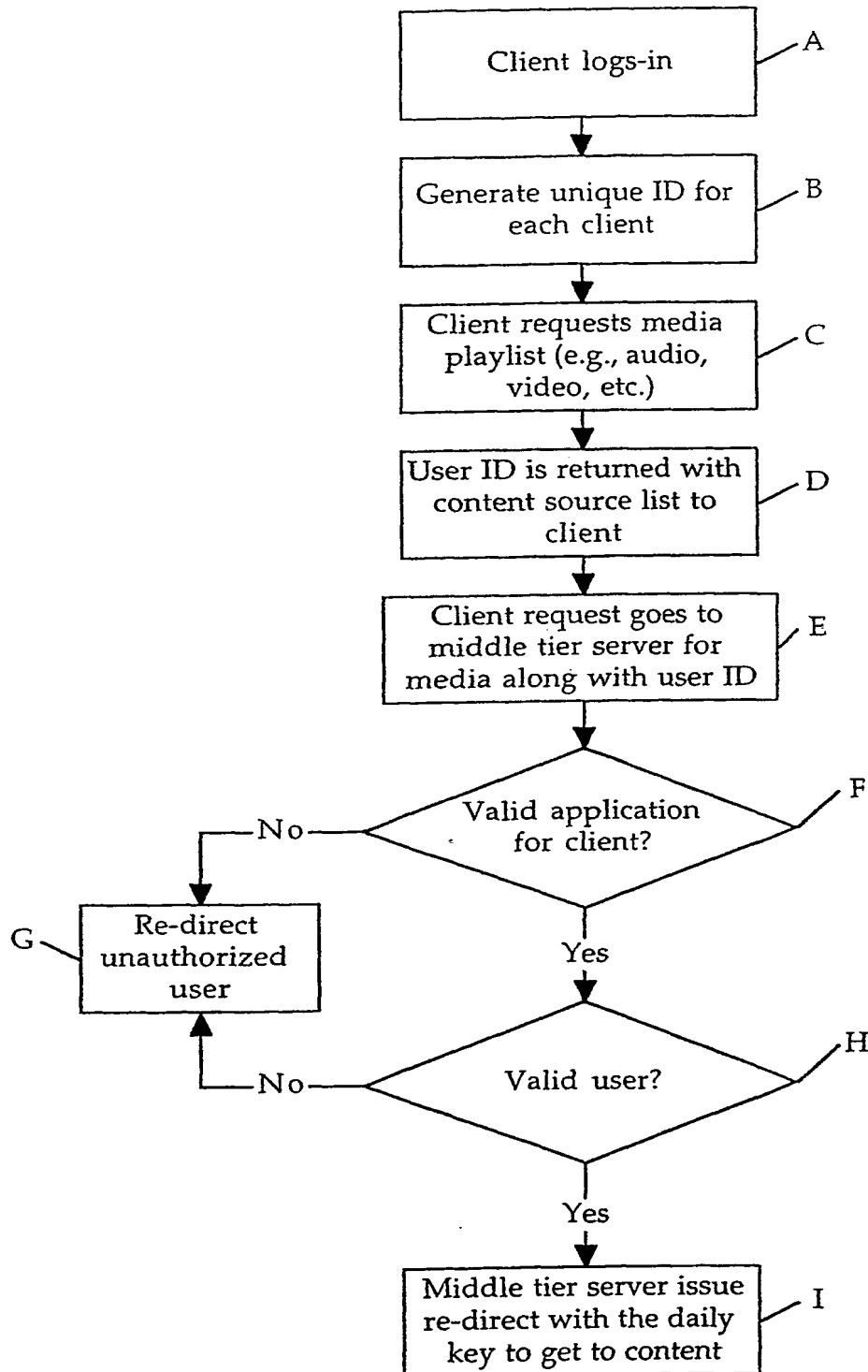
1-3000

Fig. 1-3

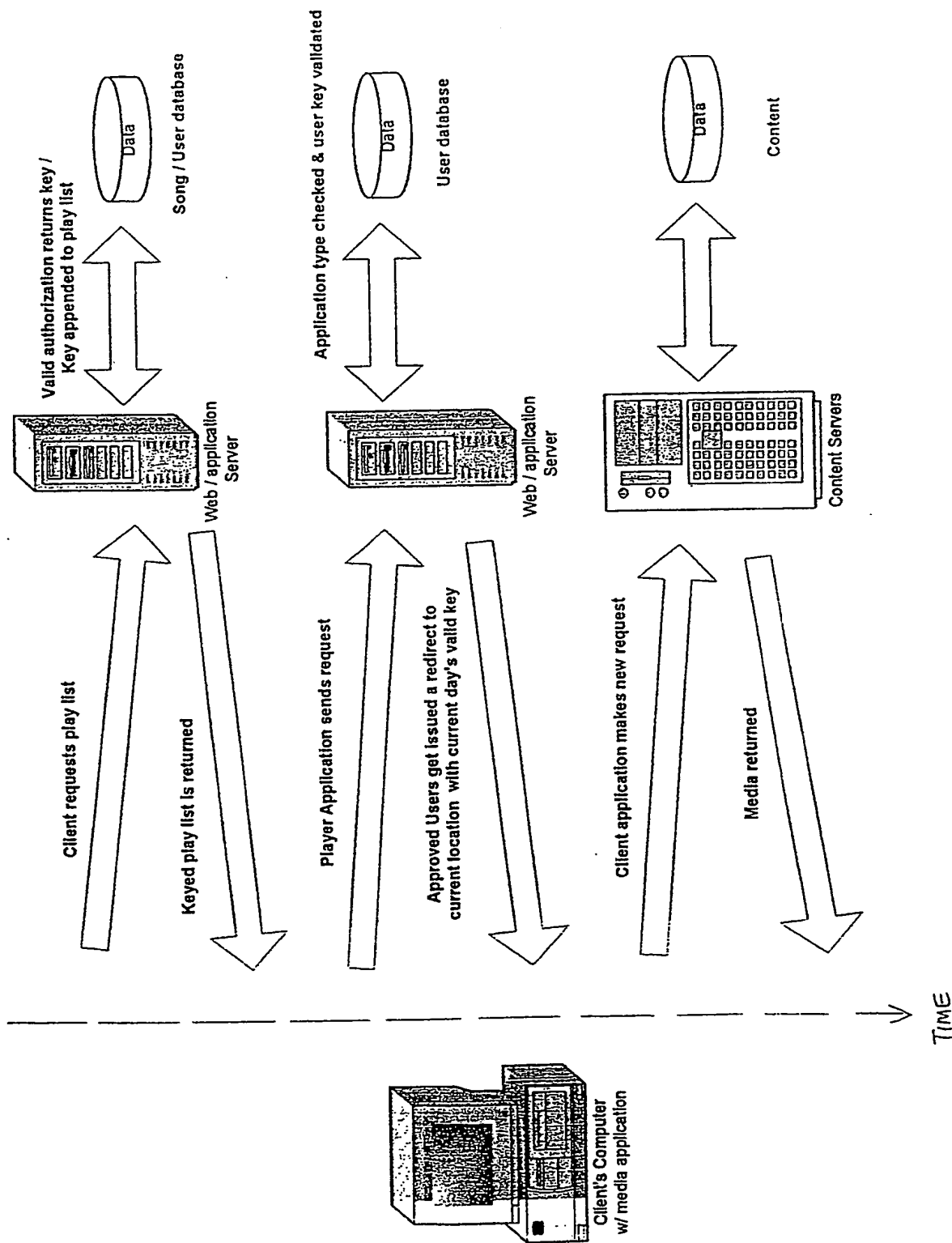


Fig. 1-4

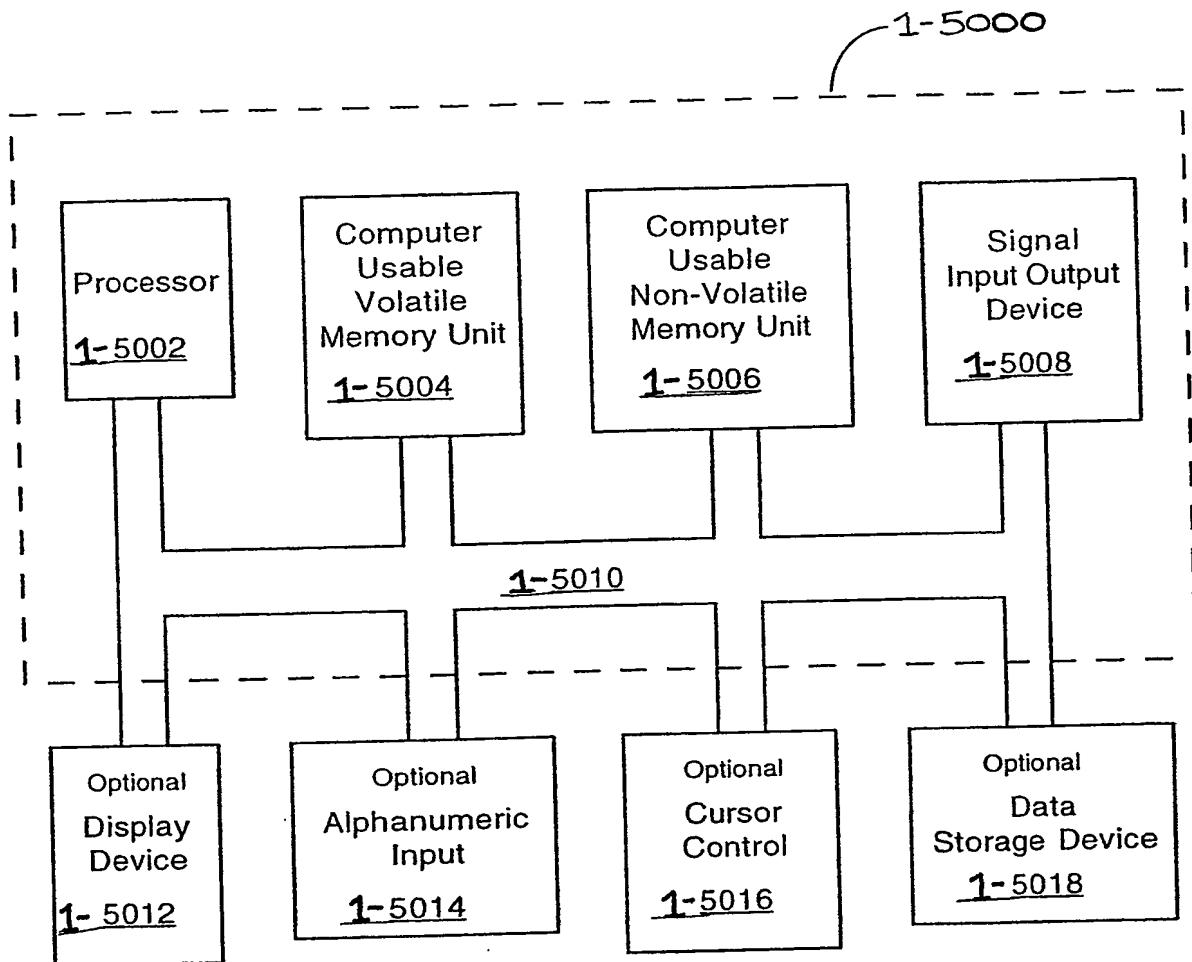
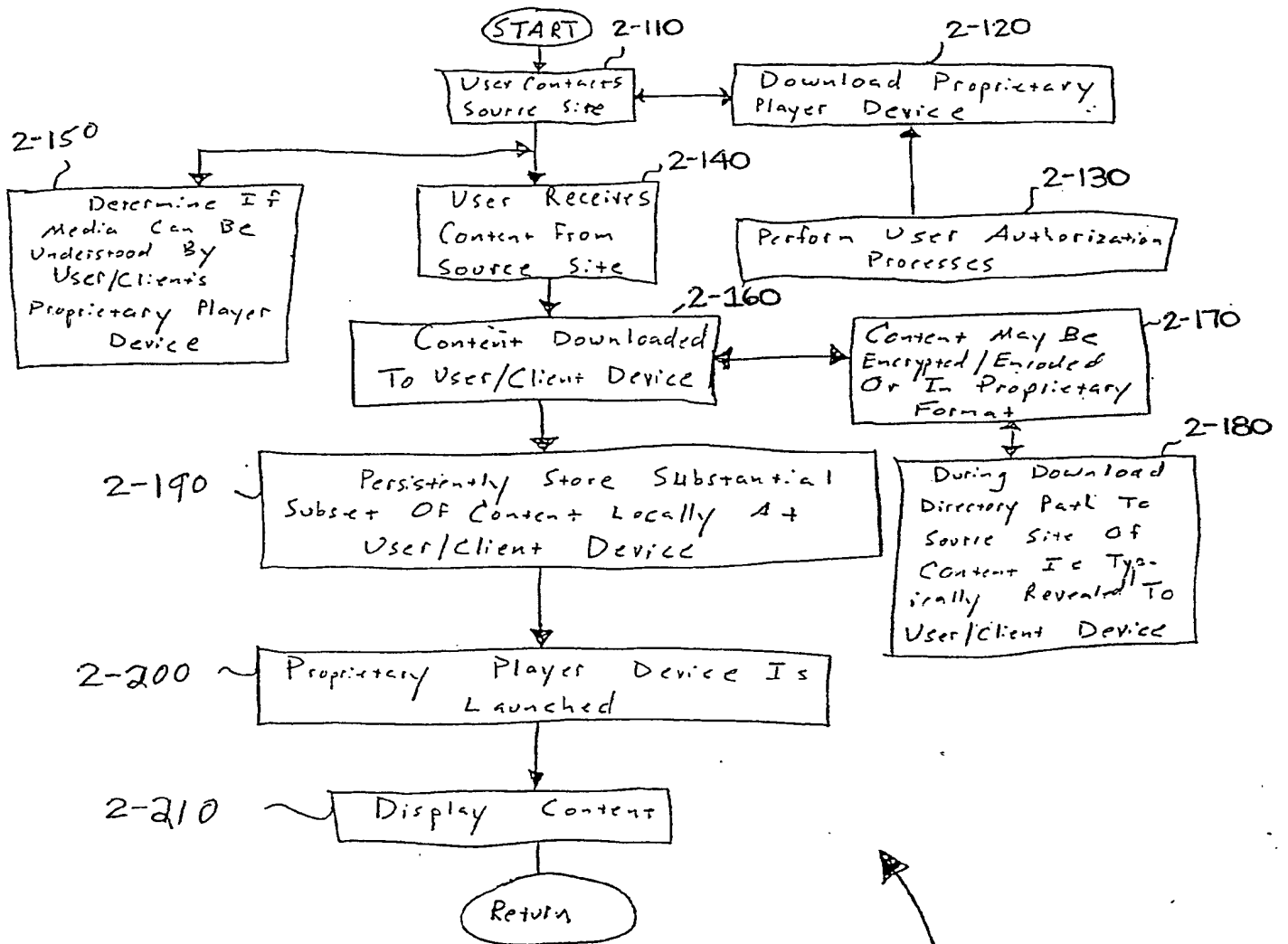


Fig. 1-5



(Prior Art)

2-100

Fig. 2-1

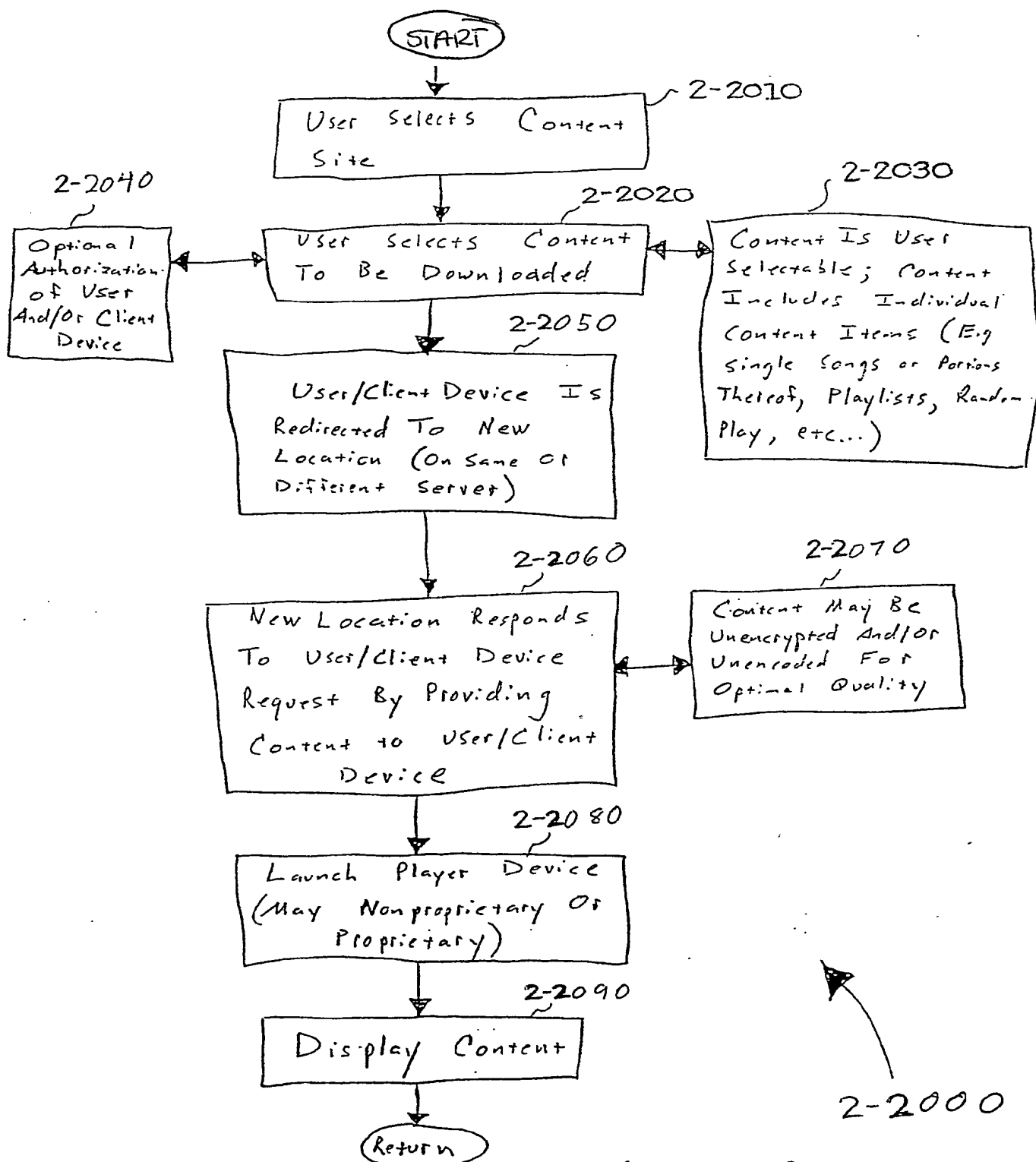


Fig. 2-2

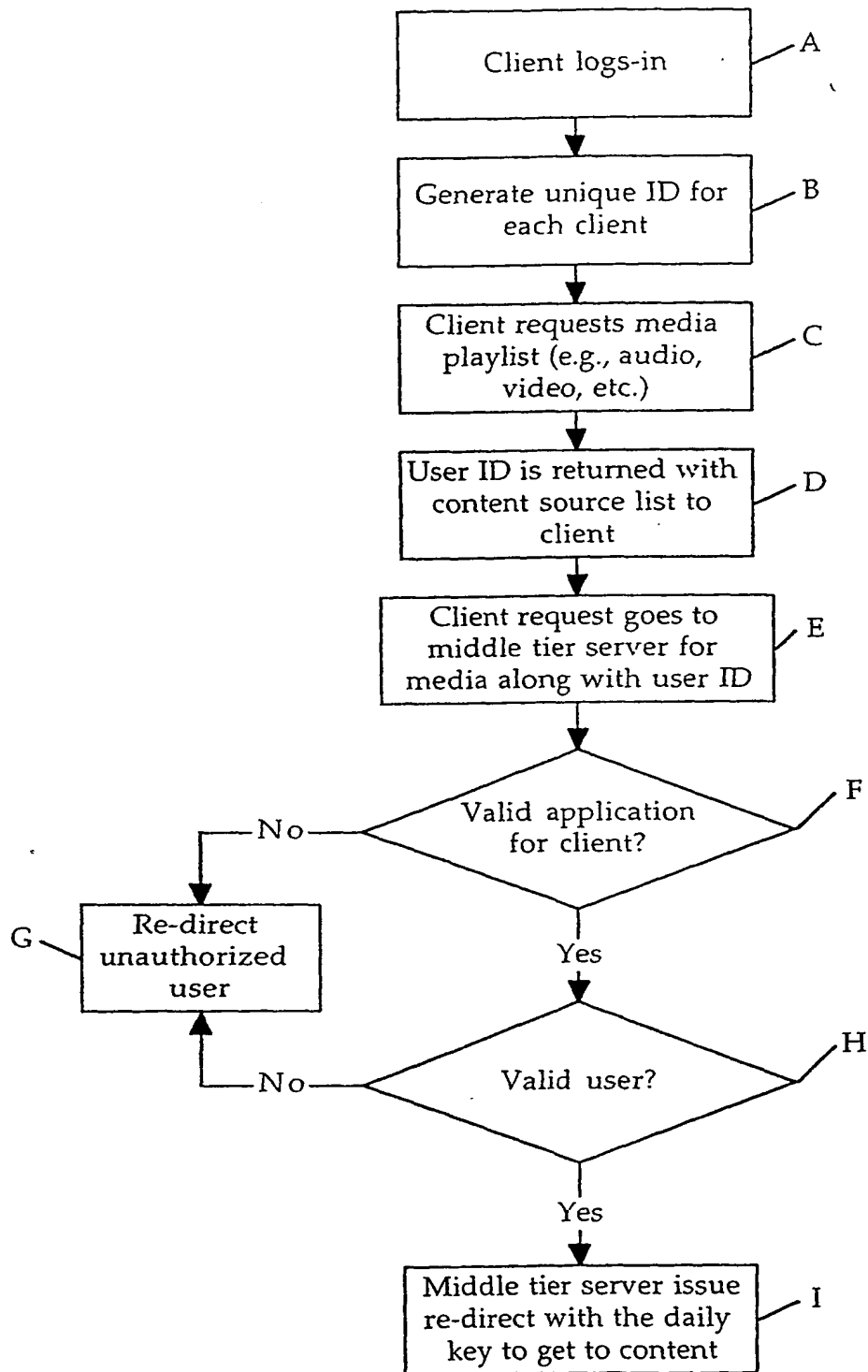


Fig. 2-3

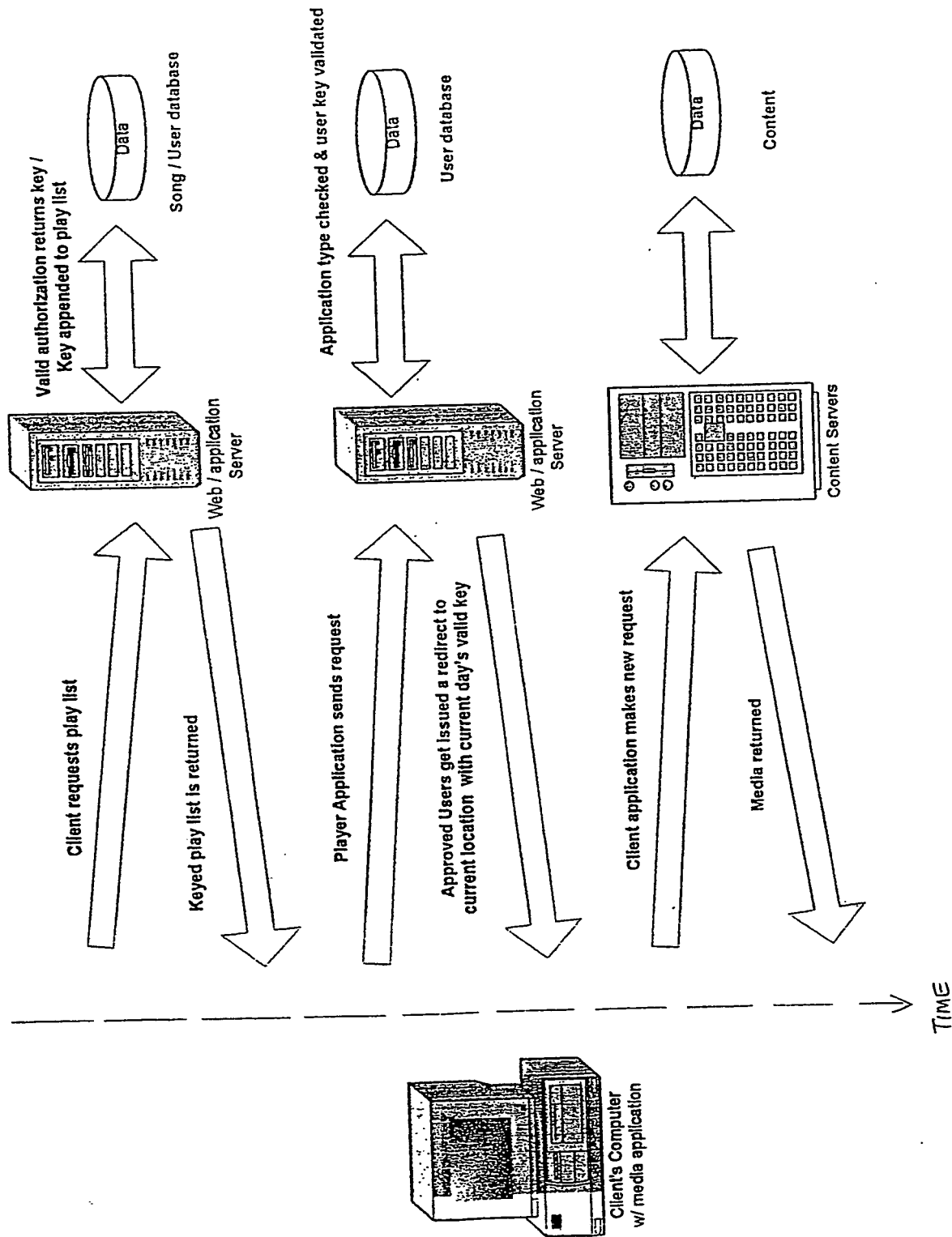


Fig. 2-4

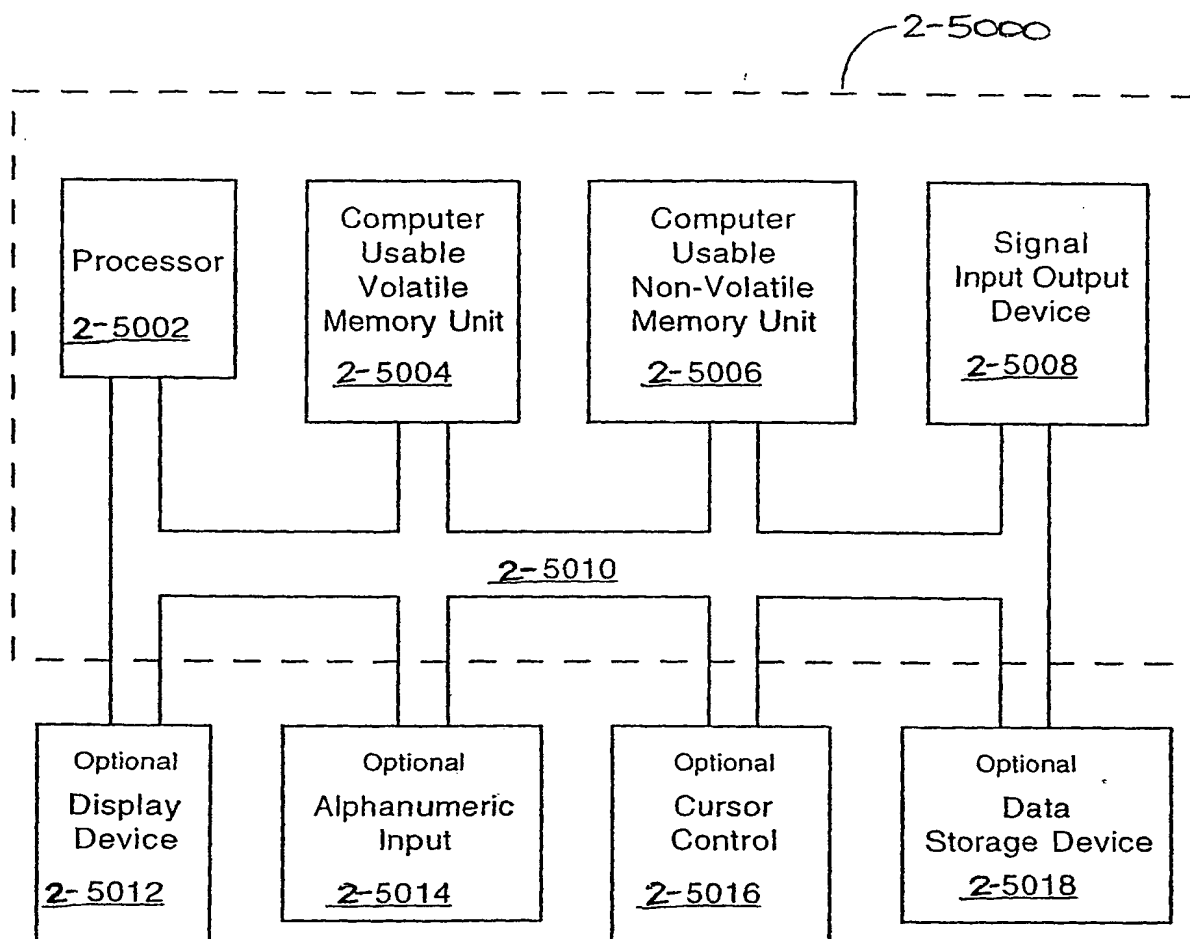


Fig. 2-5

2-6000

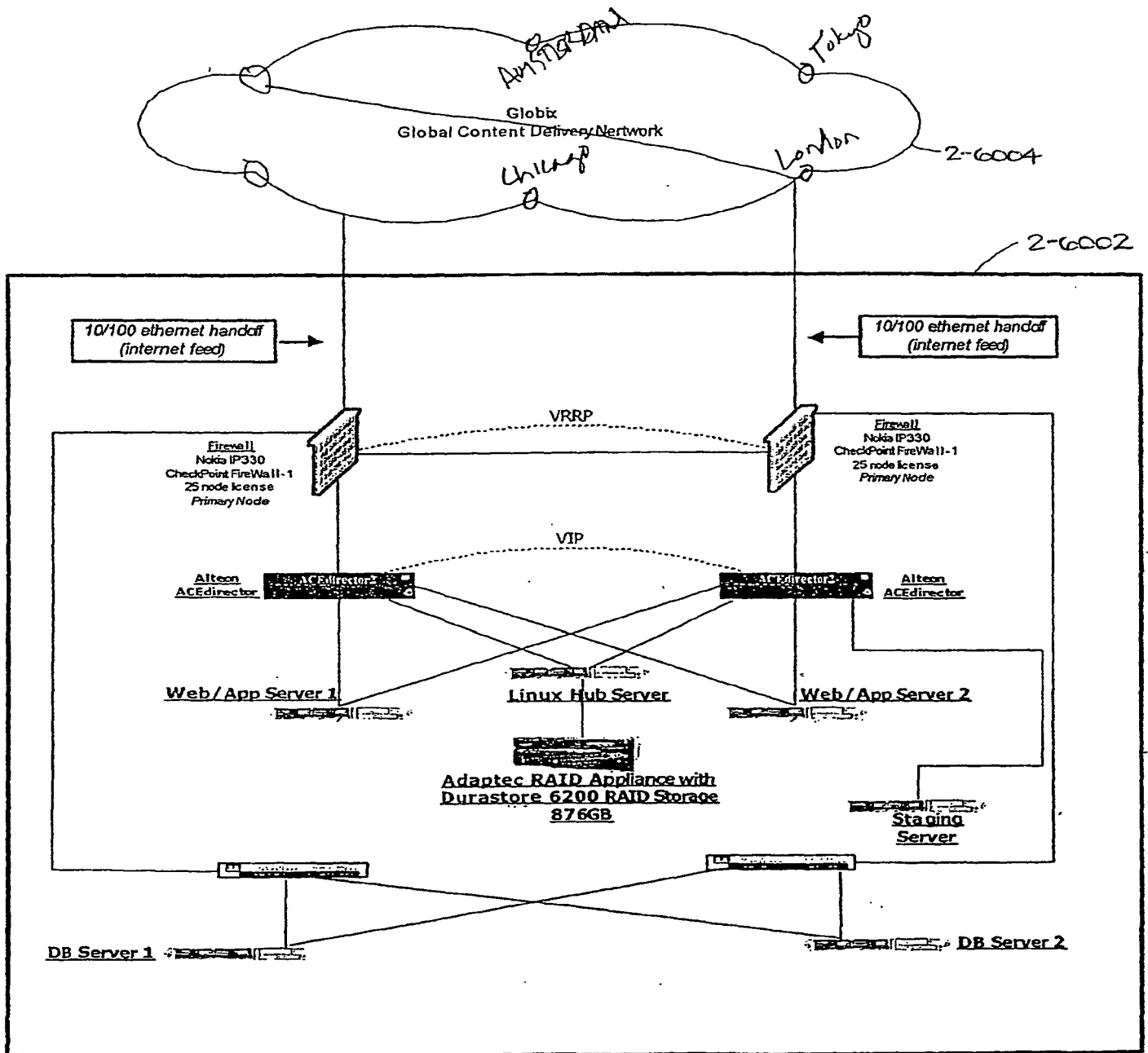


Fig. 2-6

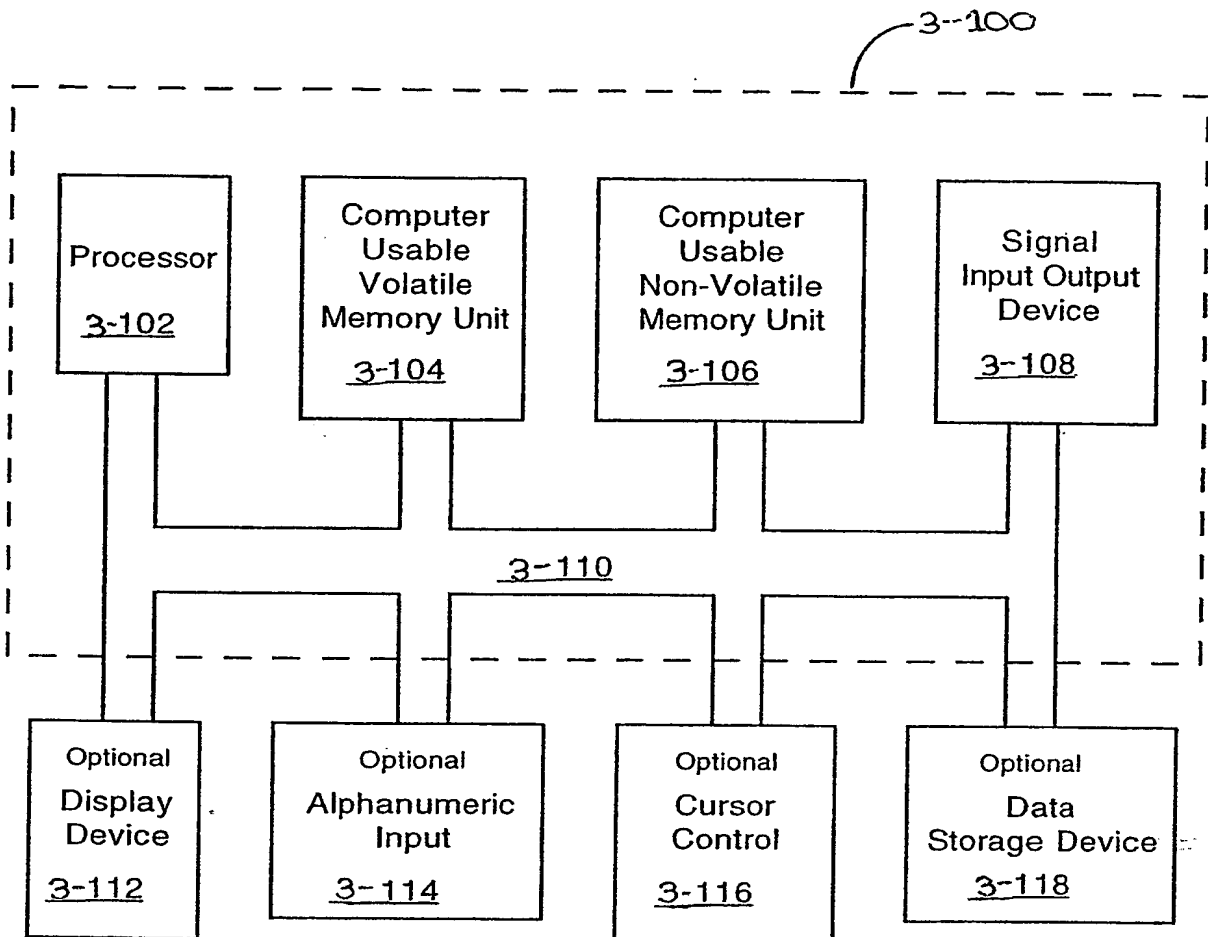


Fig. 3-1

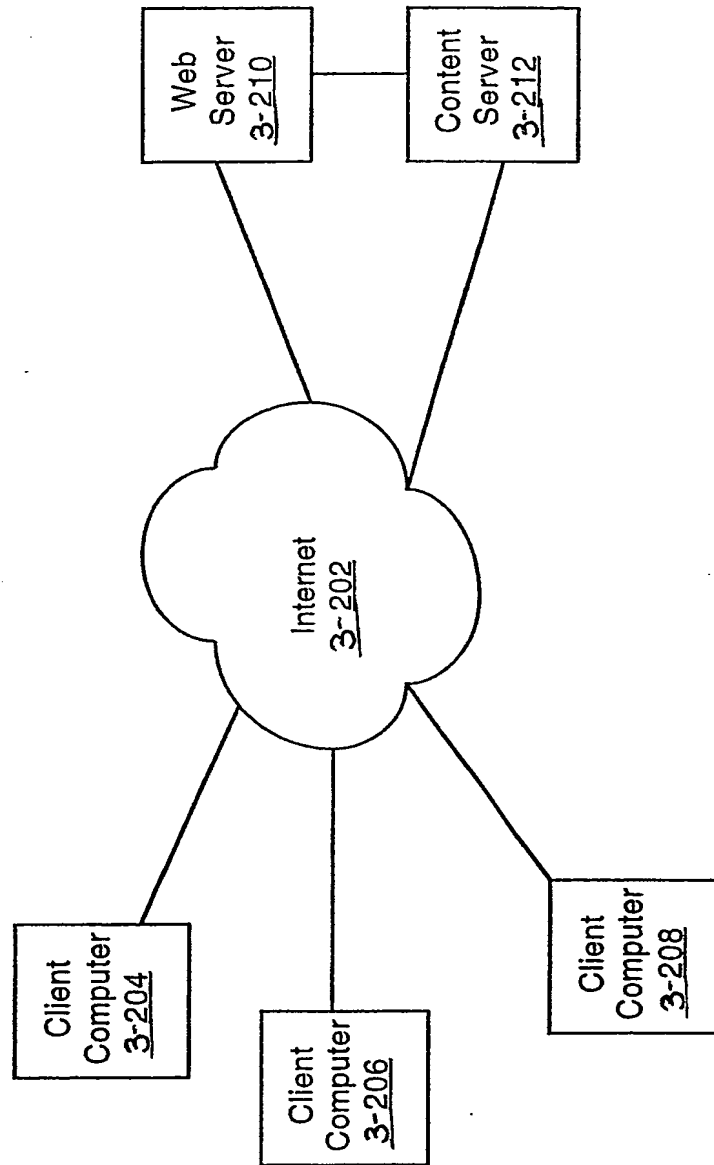


Fig. 3-2

3-200

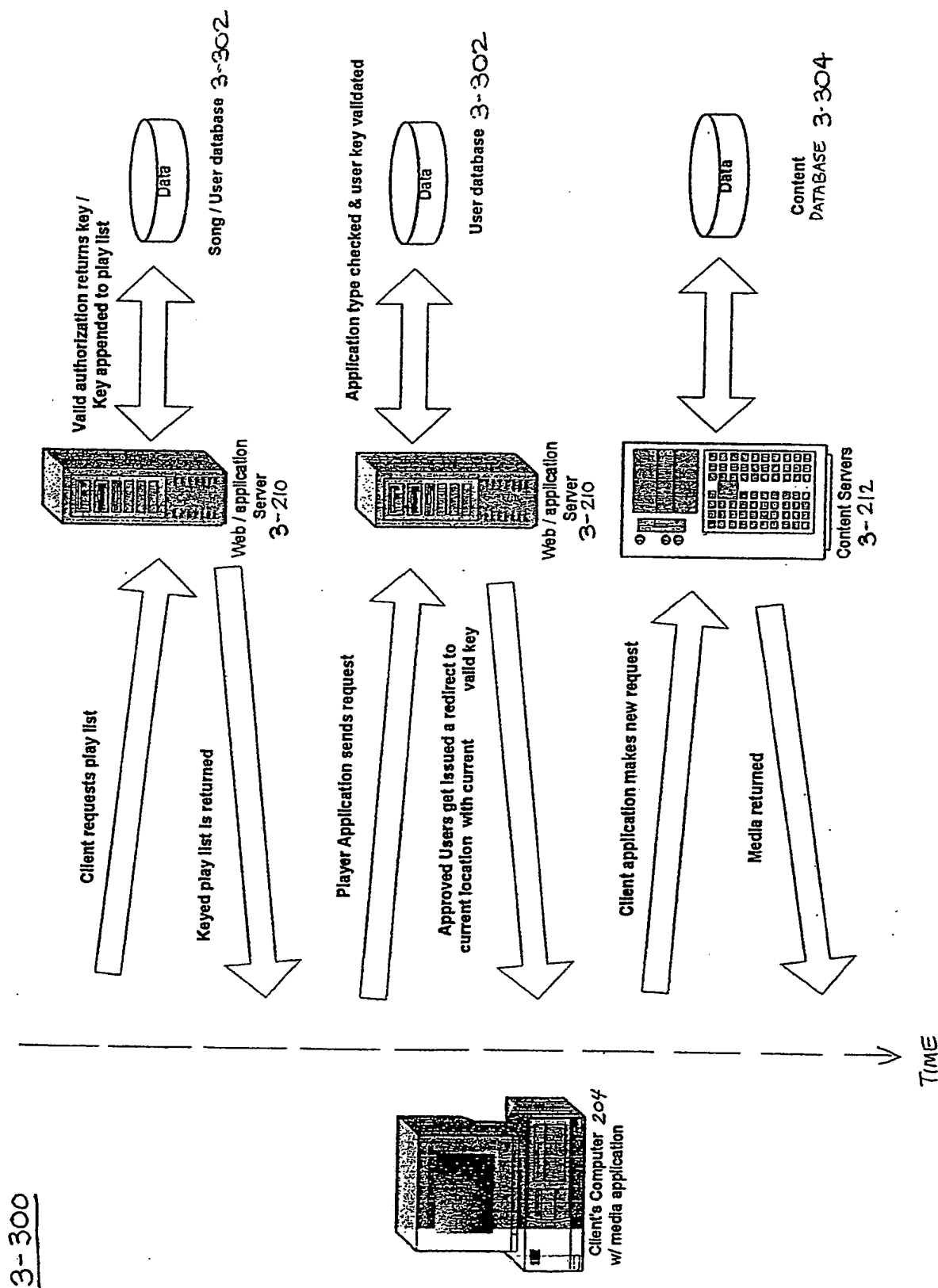


Fig. 3-3

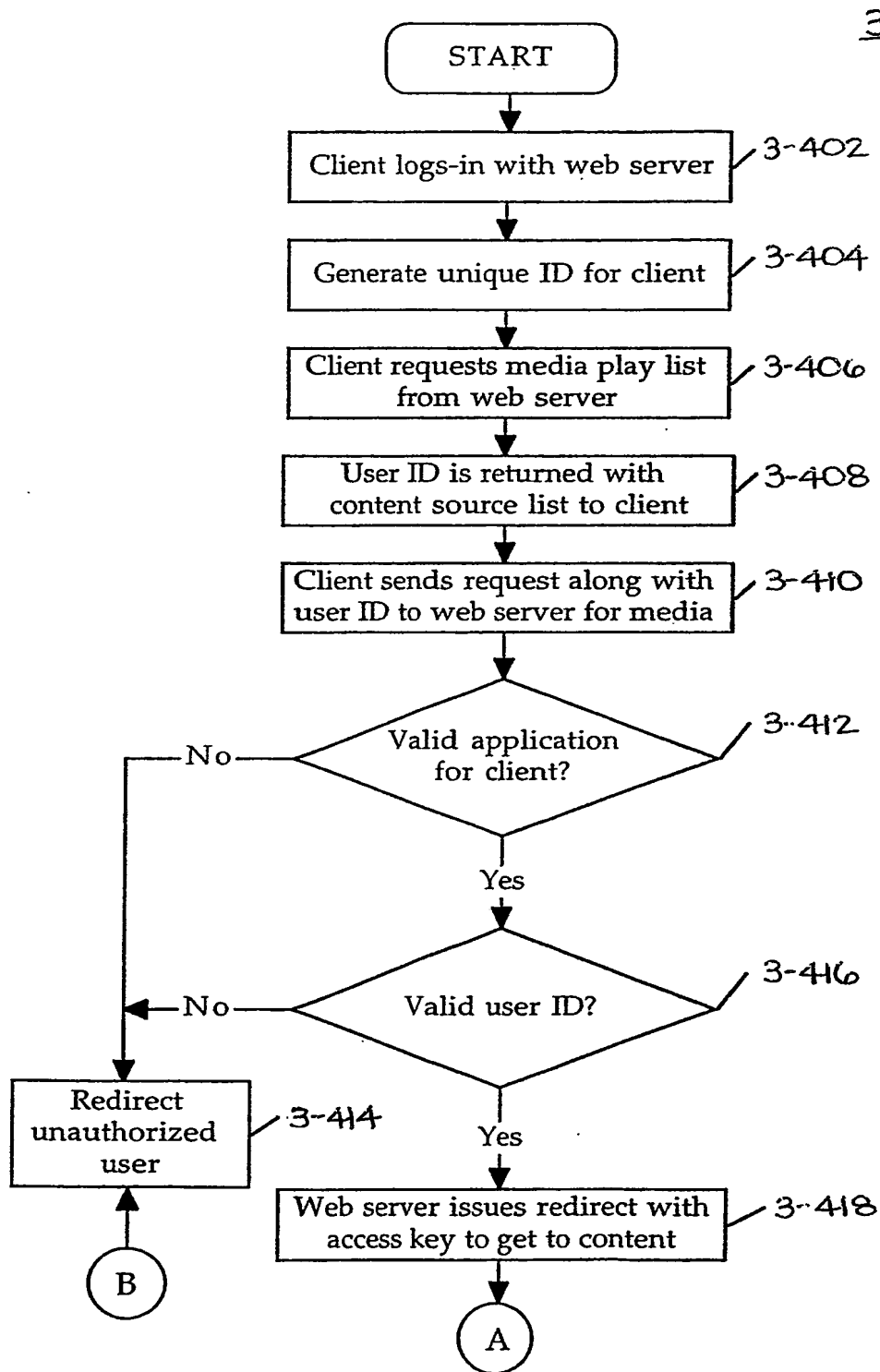


Fig. 3-4A

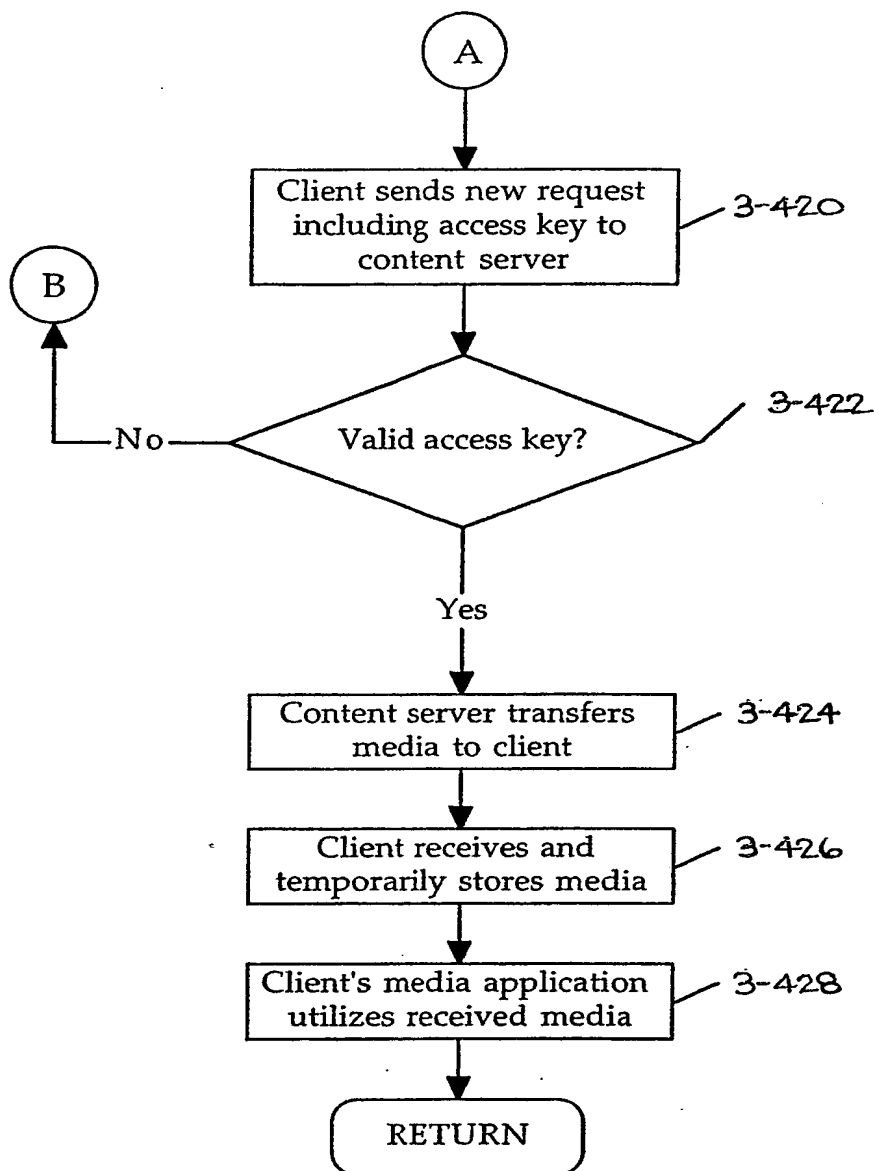
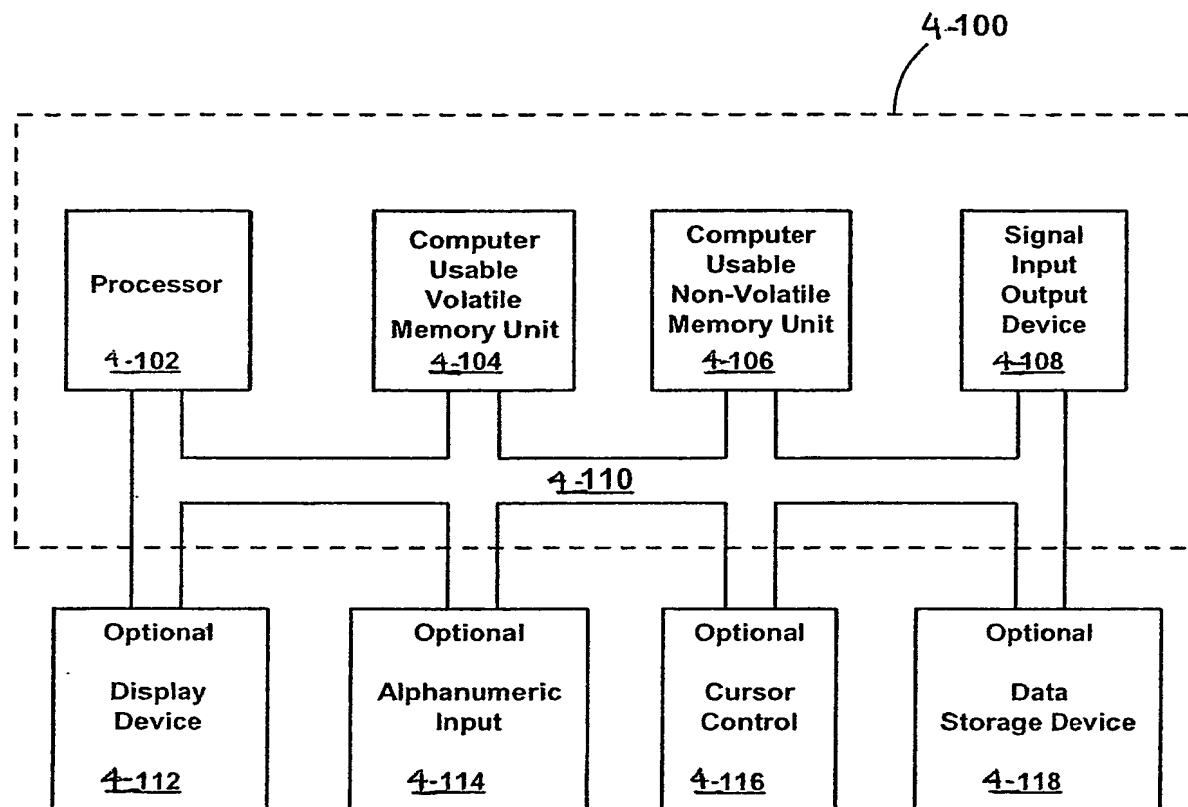


Fig. 3-4B

**FIG. 4-1**

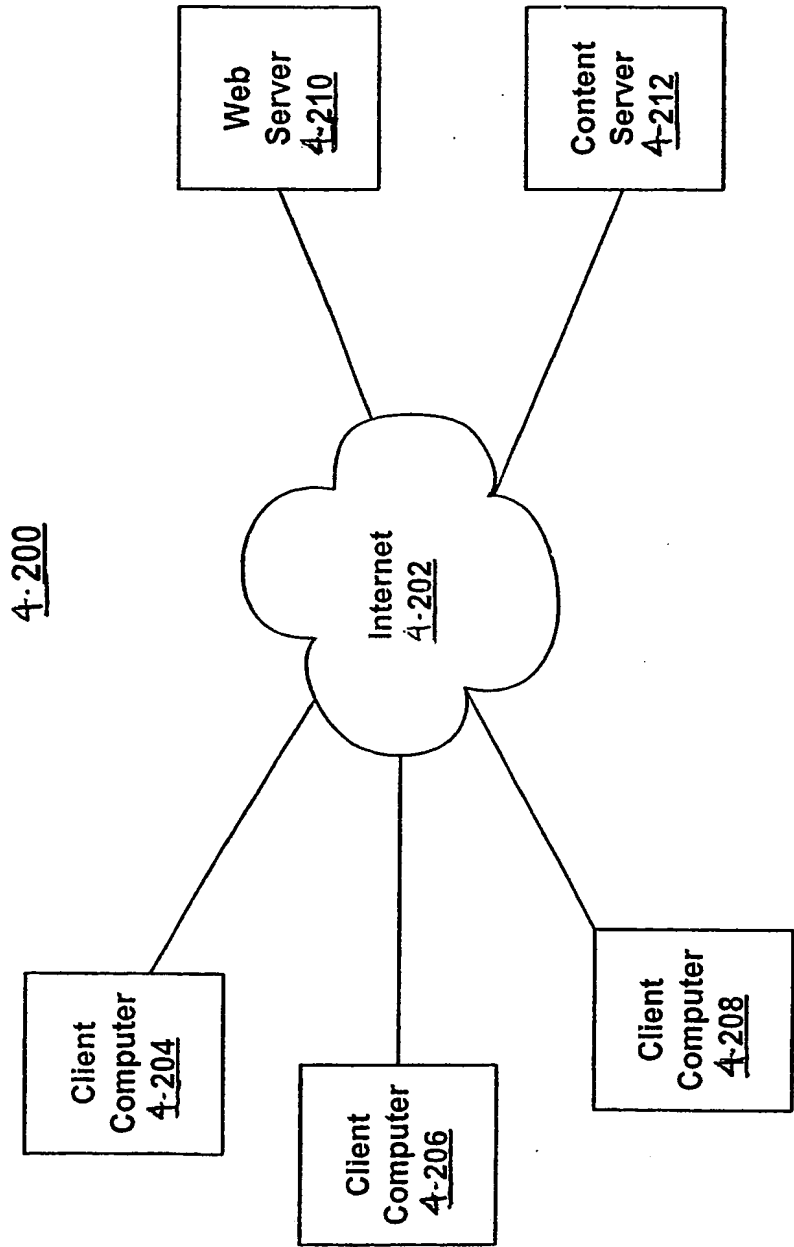


FIG. 4-2

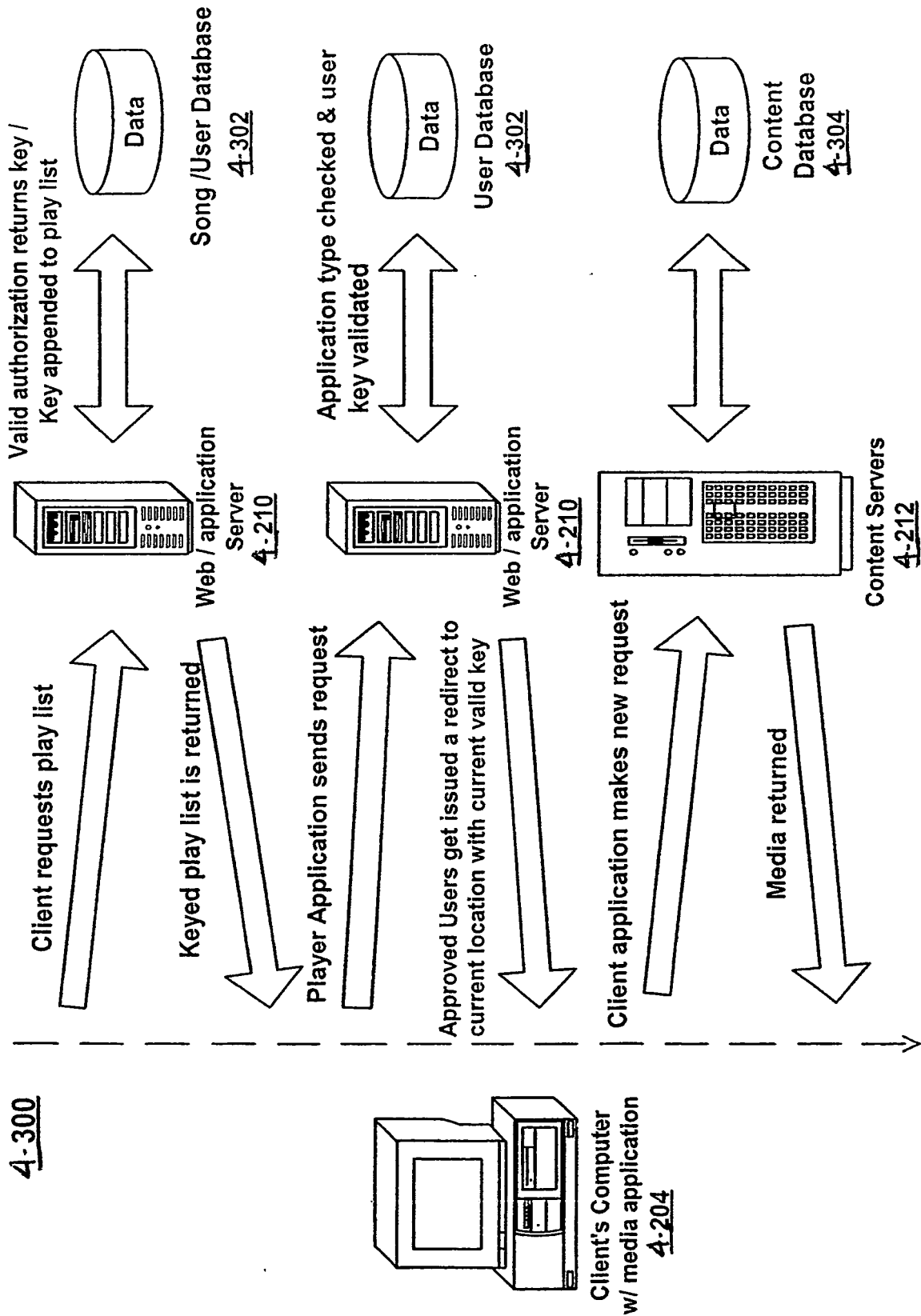


FIG. 4-3

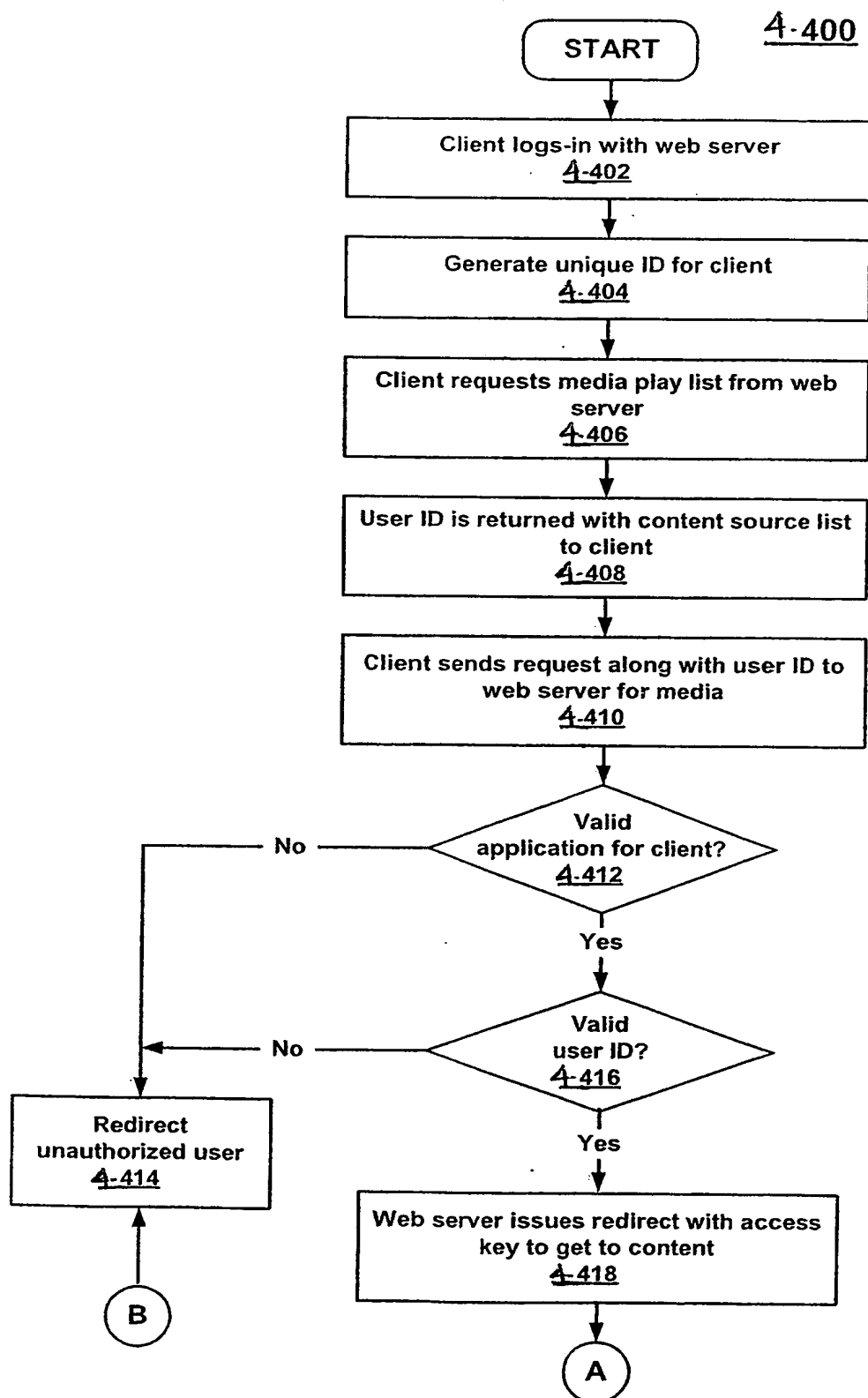


FIG. 4-4A

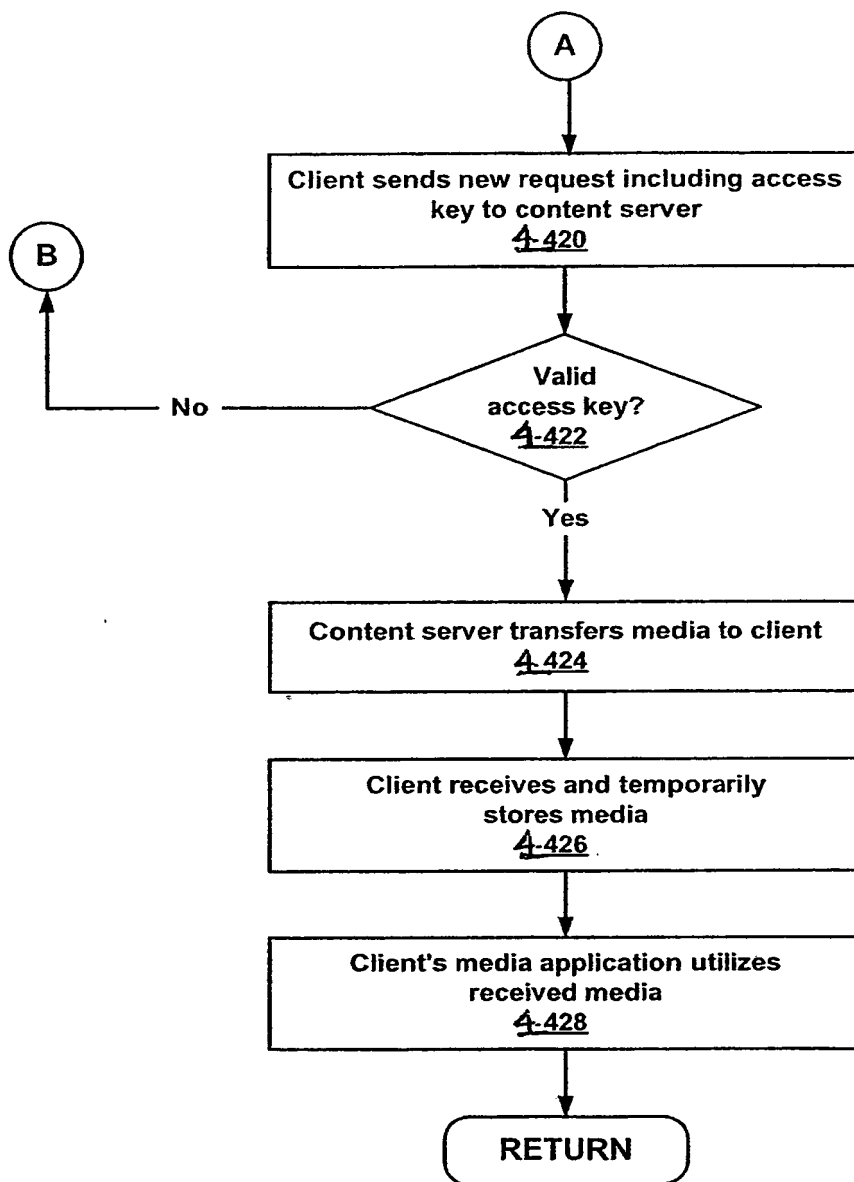


FIG. 4-4B

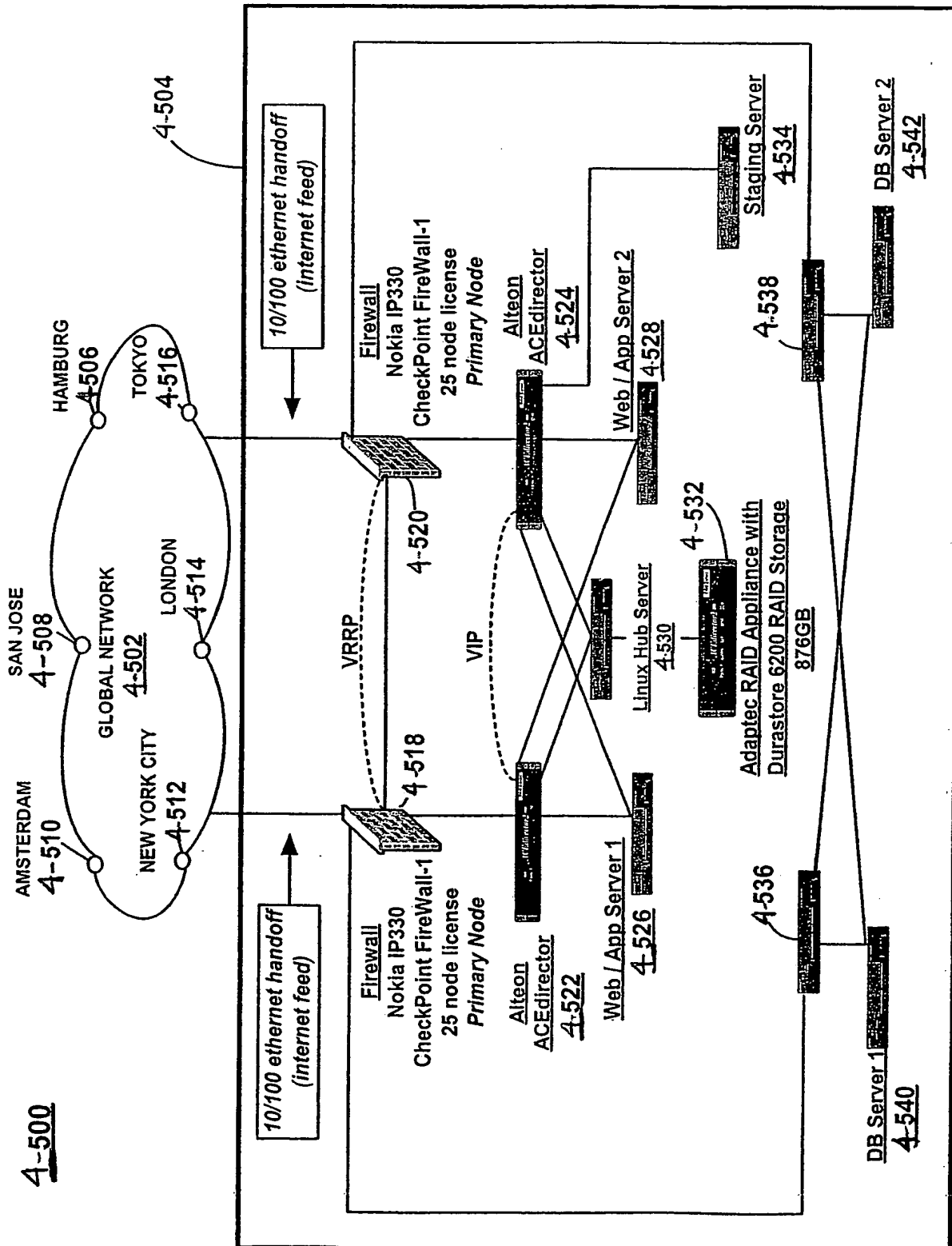


FIG. 4-5

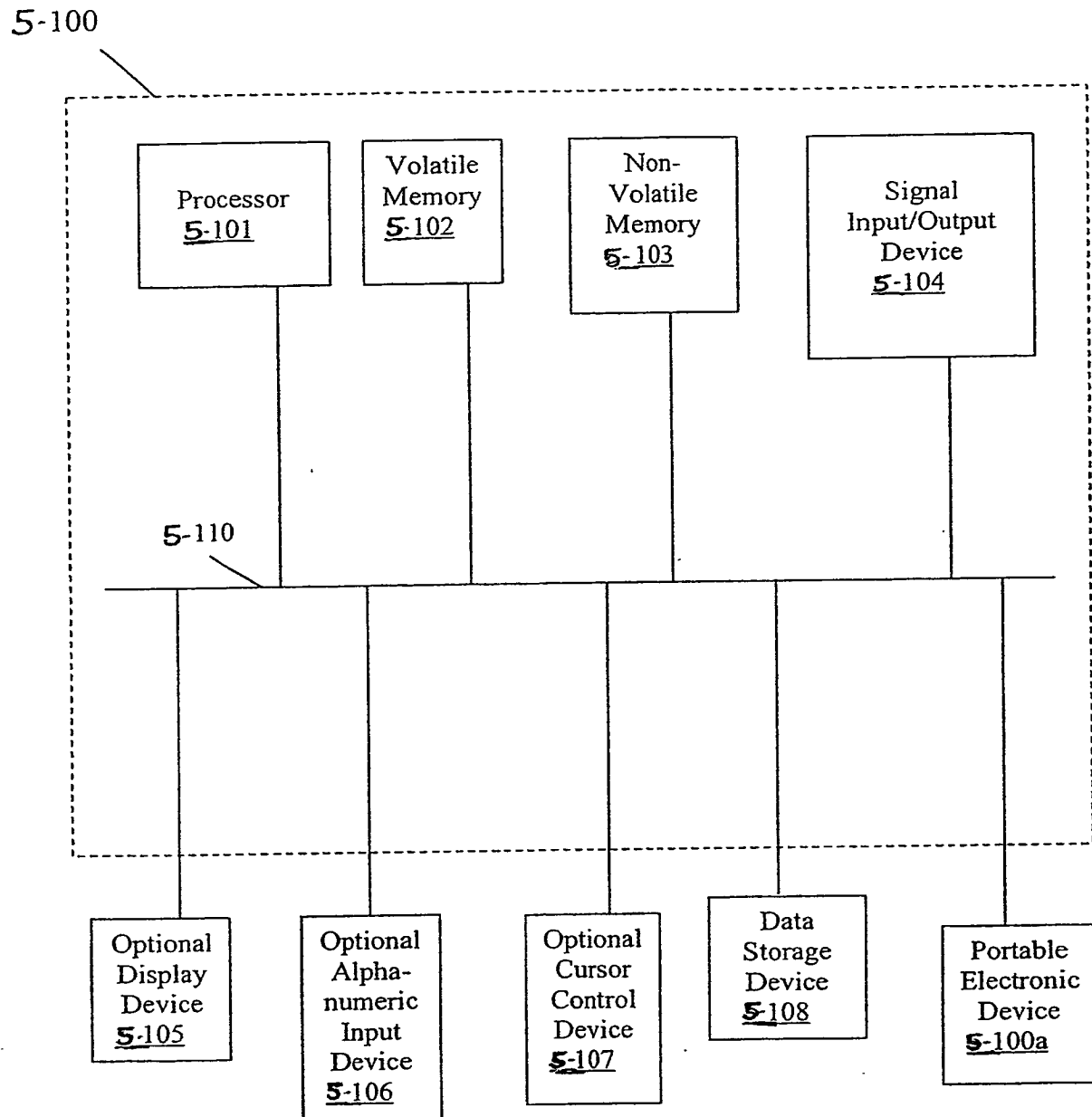


FIGURE 5-1

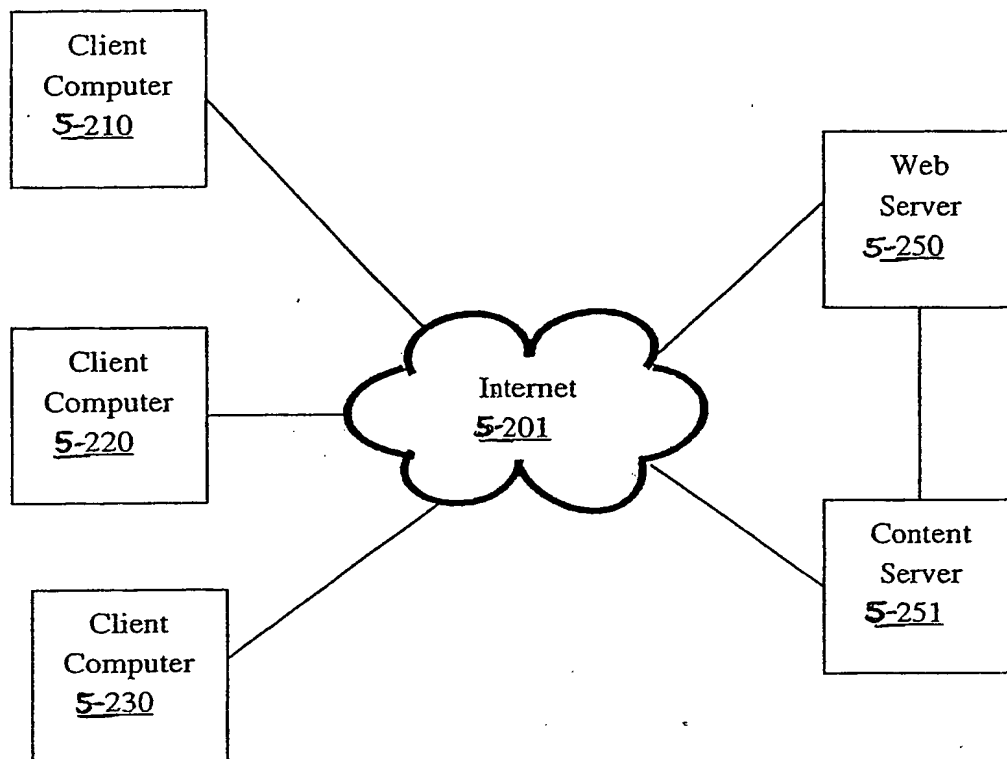
5-200

FIGURE 5-2

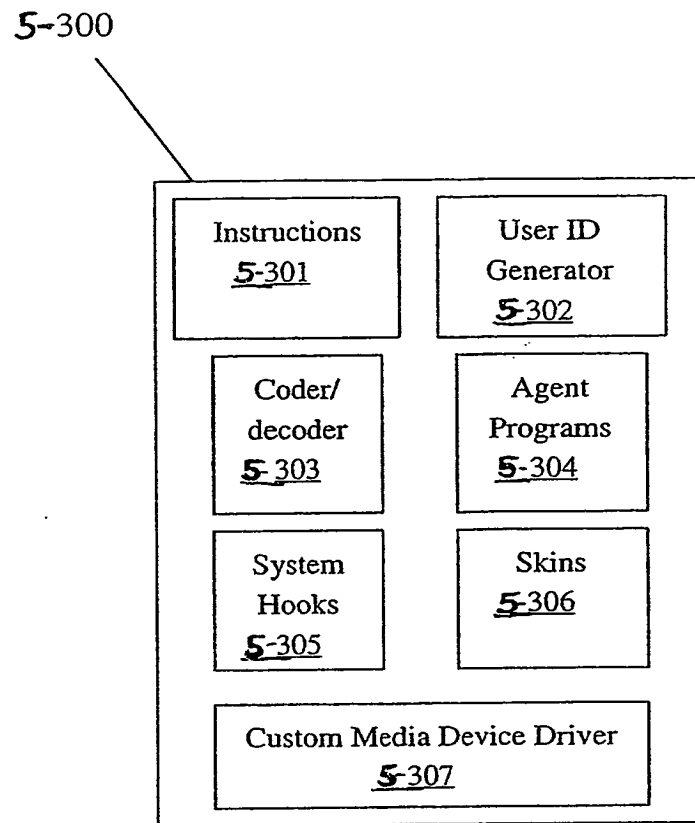


FIGURE 5-3

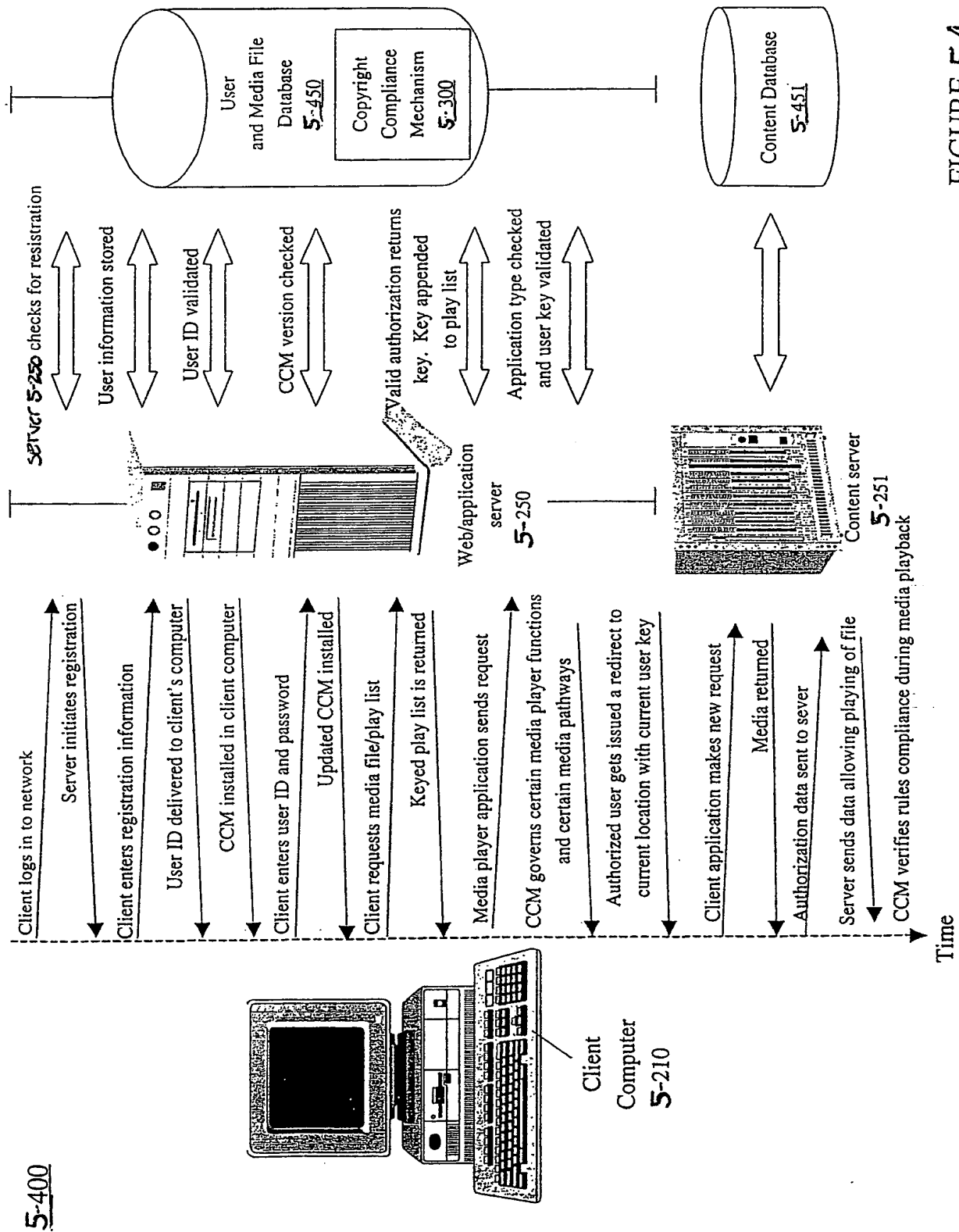


FIGURE 5-4

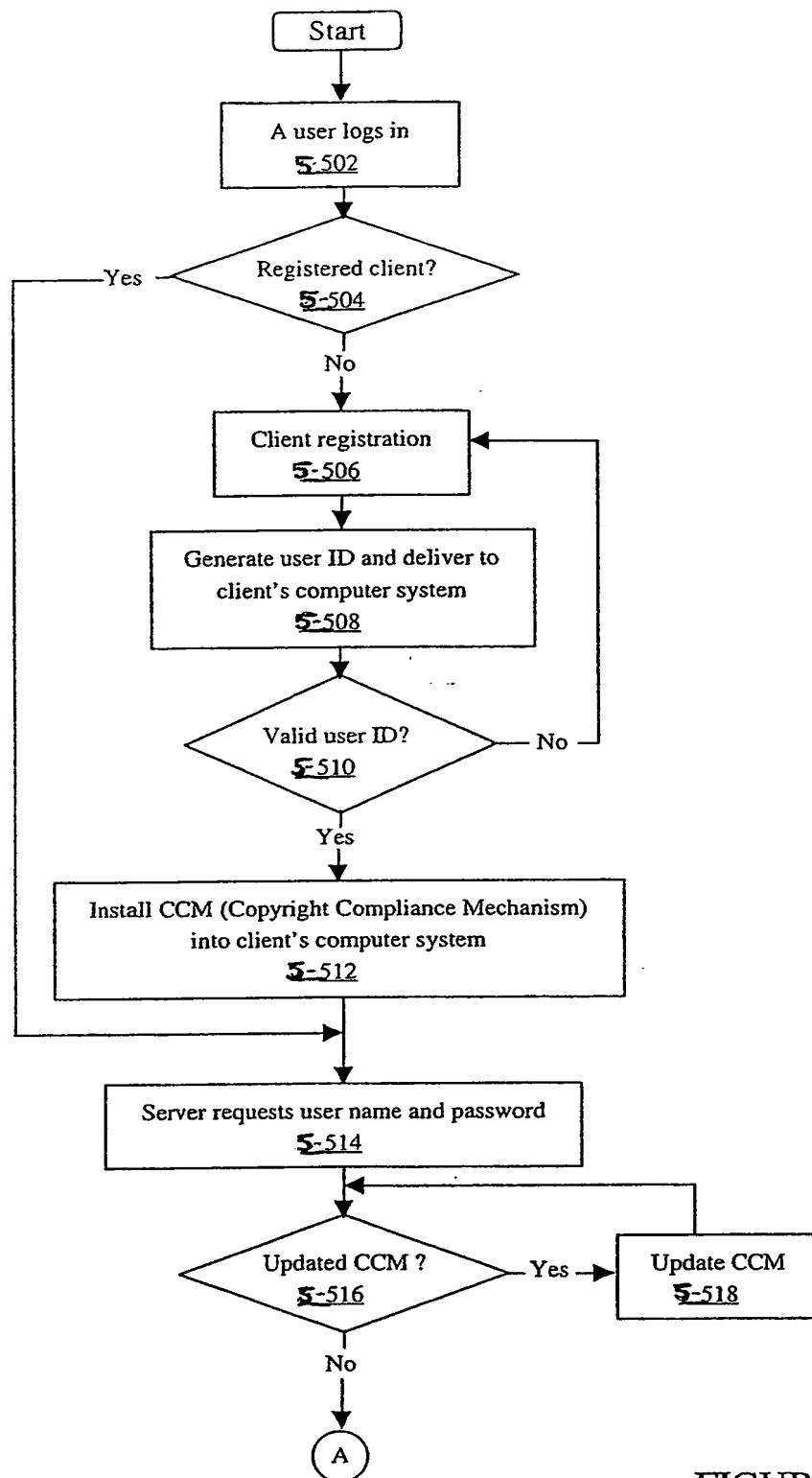
5-500

FIGURE 5-5A

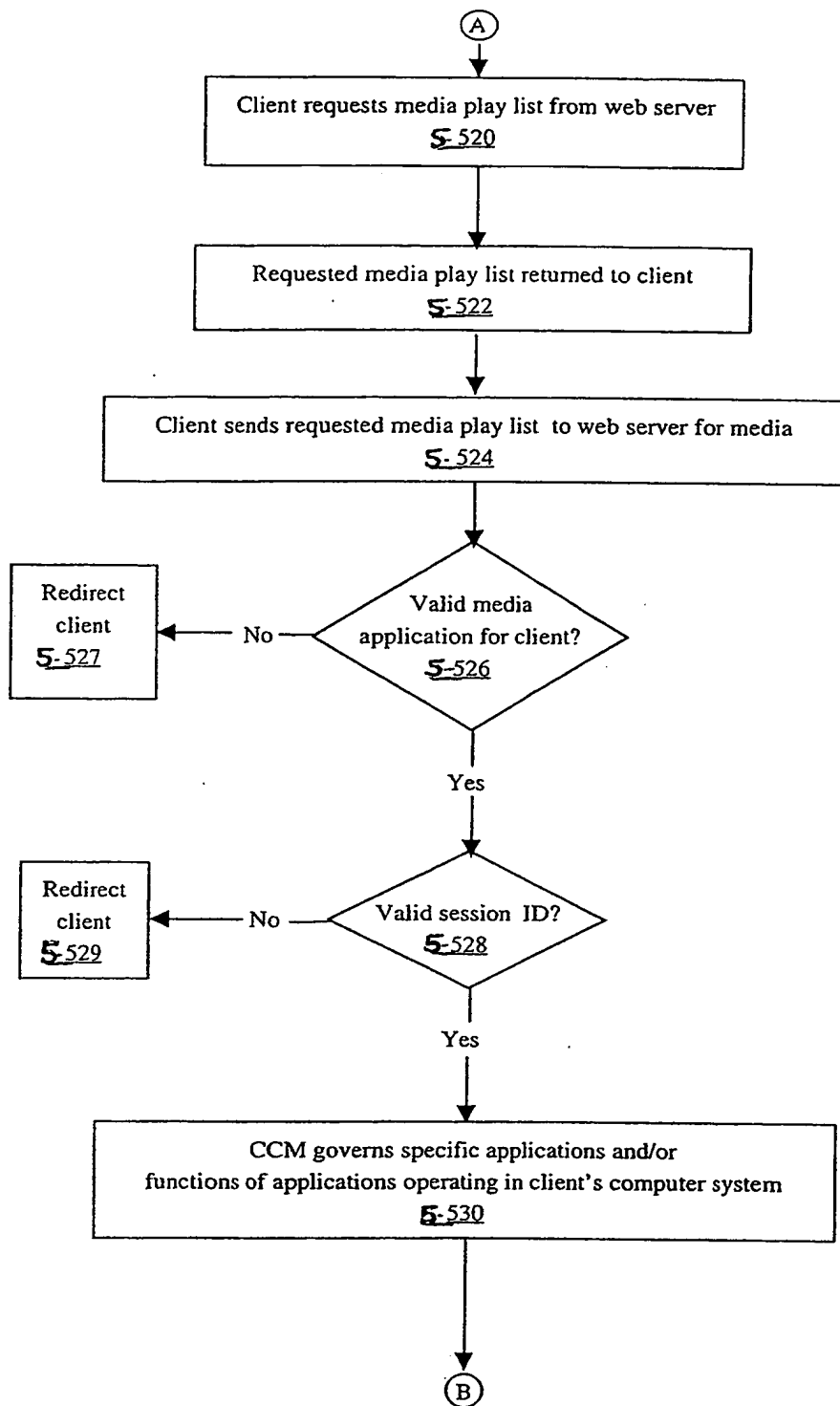
5-500

FIGURE 5-5B

45/108

5-500

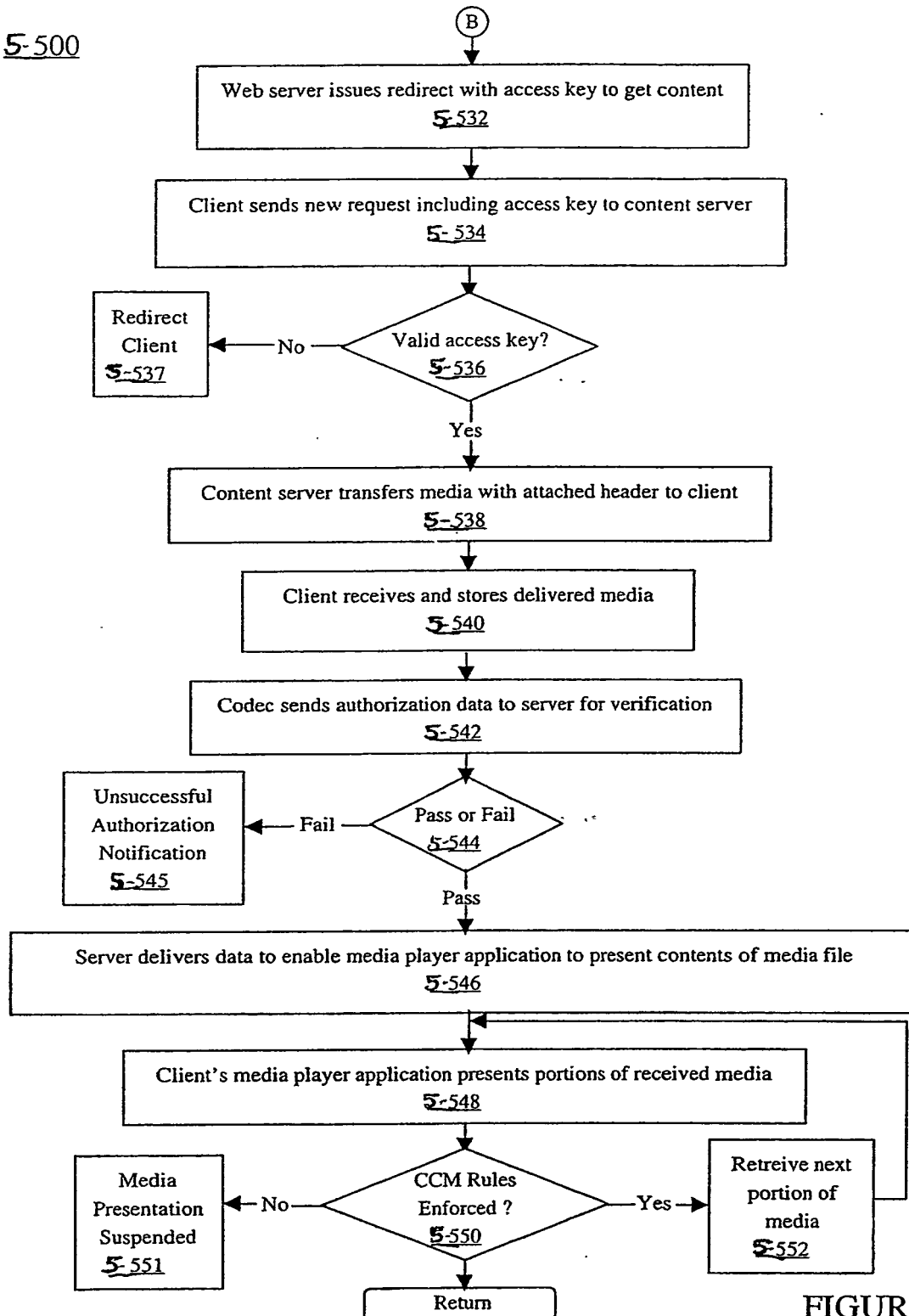


FIGURE 5-5C

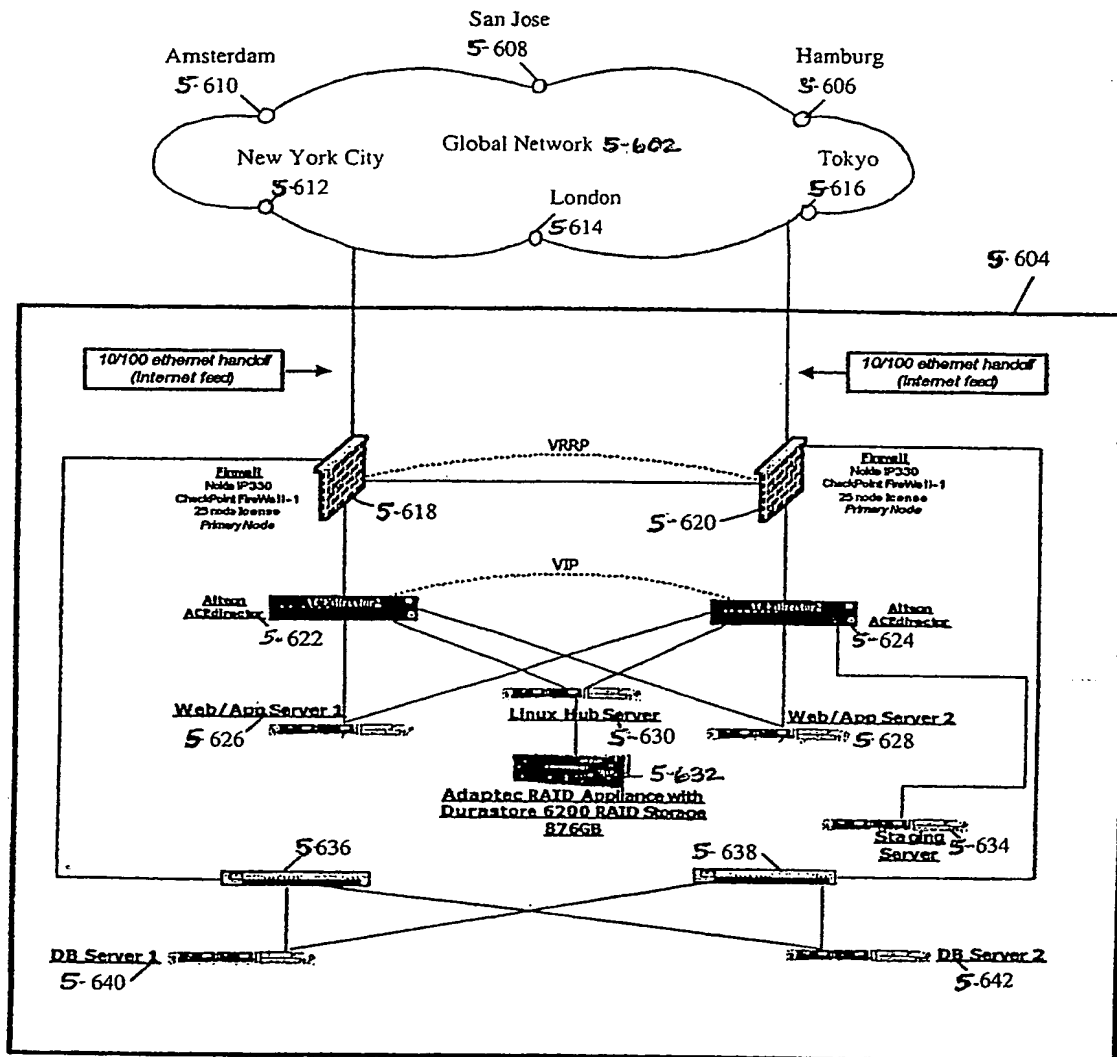
5-600

FIGURE 5-6

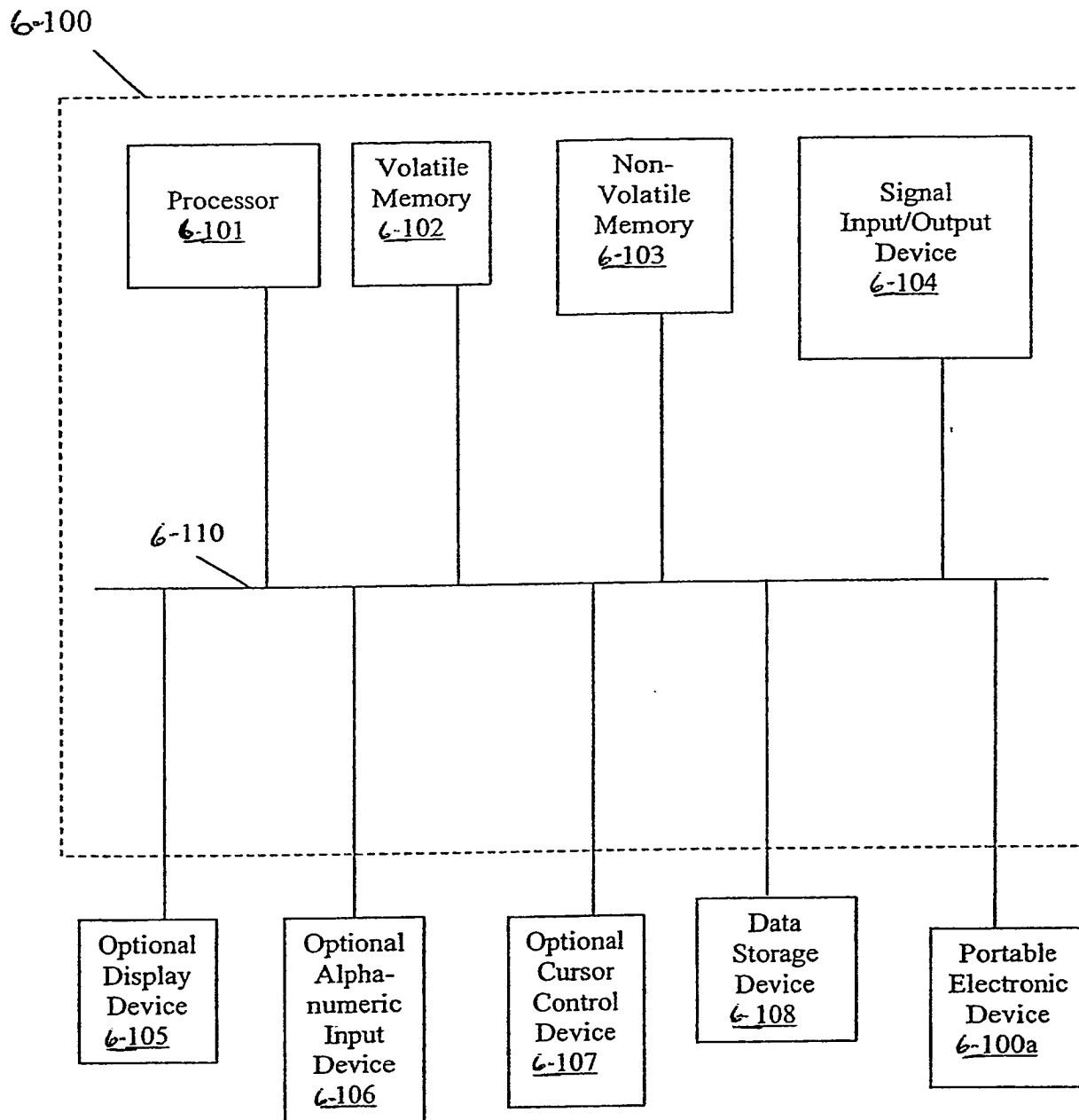


FIGURE 6-1

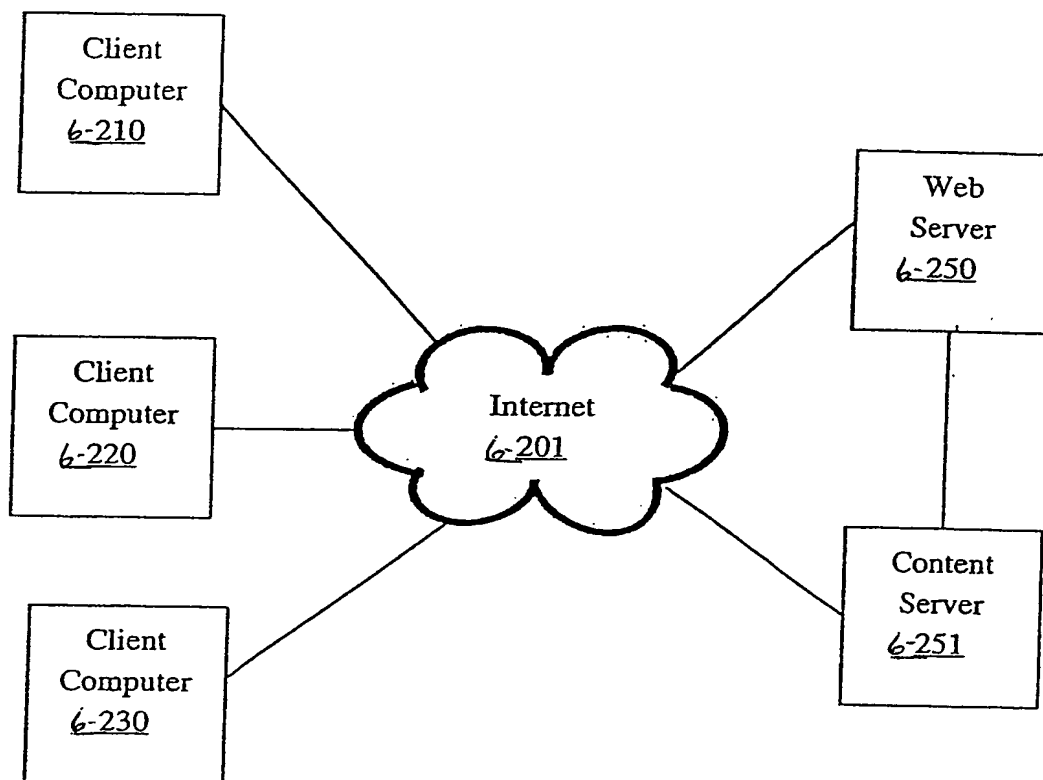
6-200

FIGURE 6-2

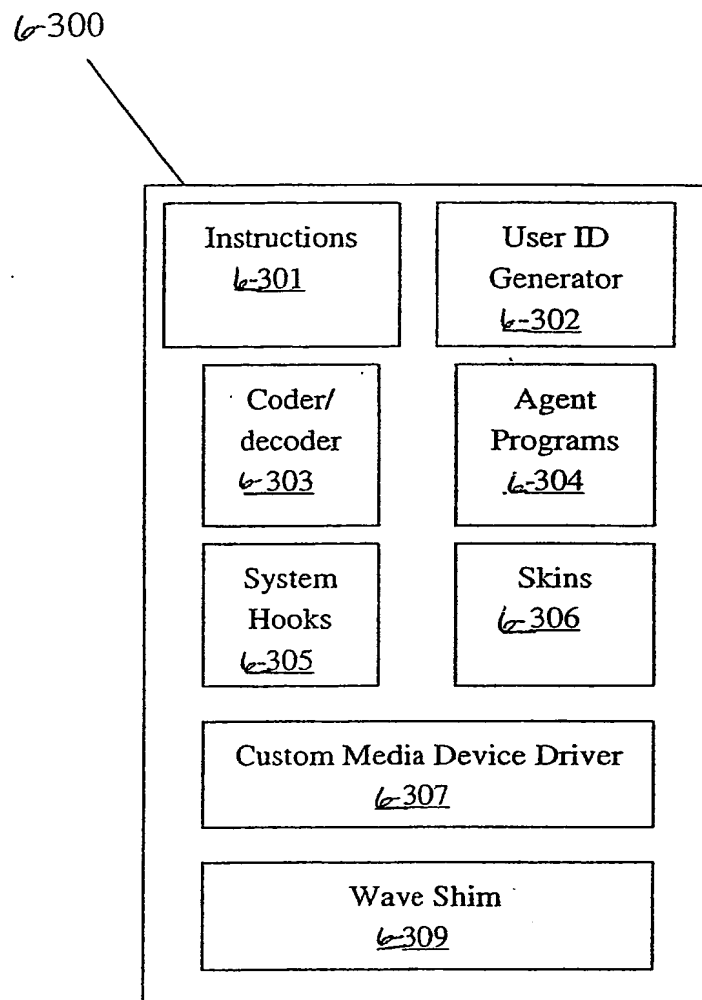


FIGURE 6-3

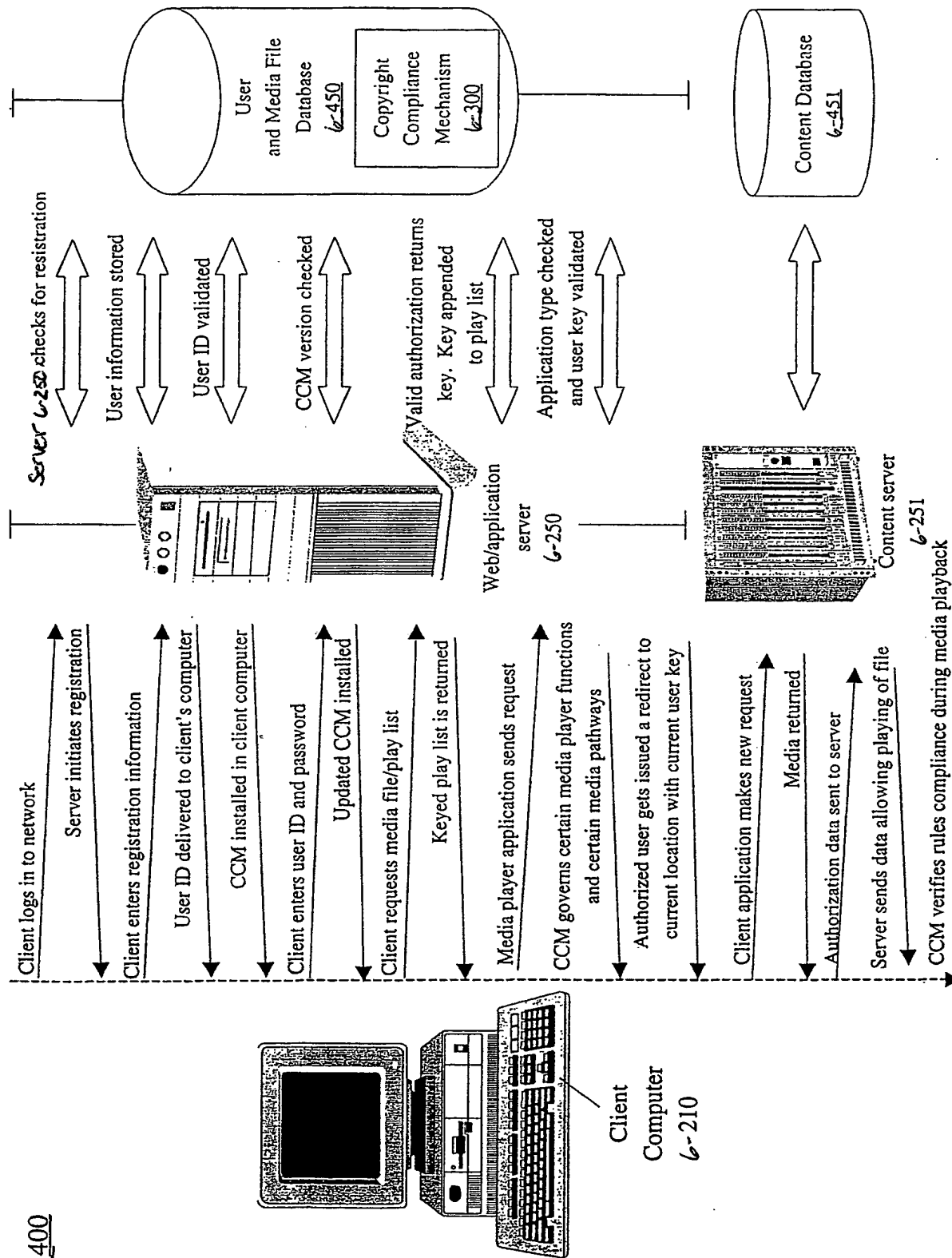
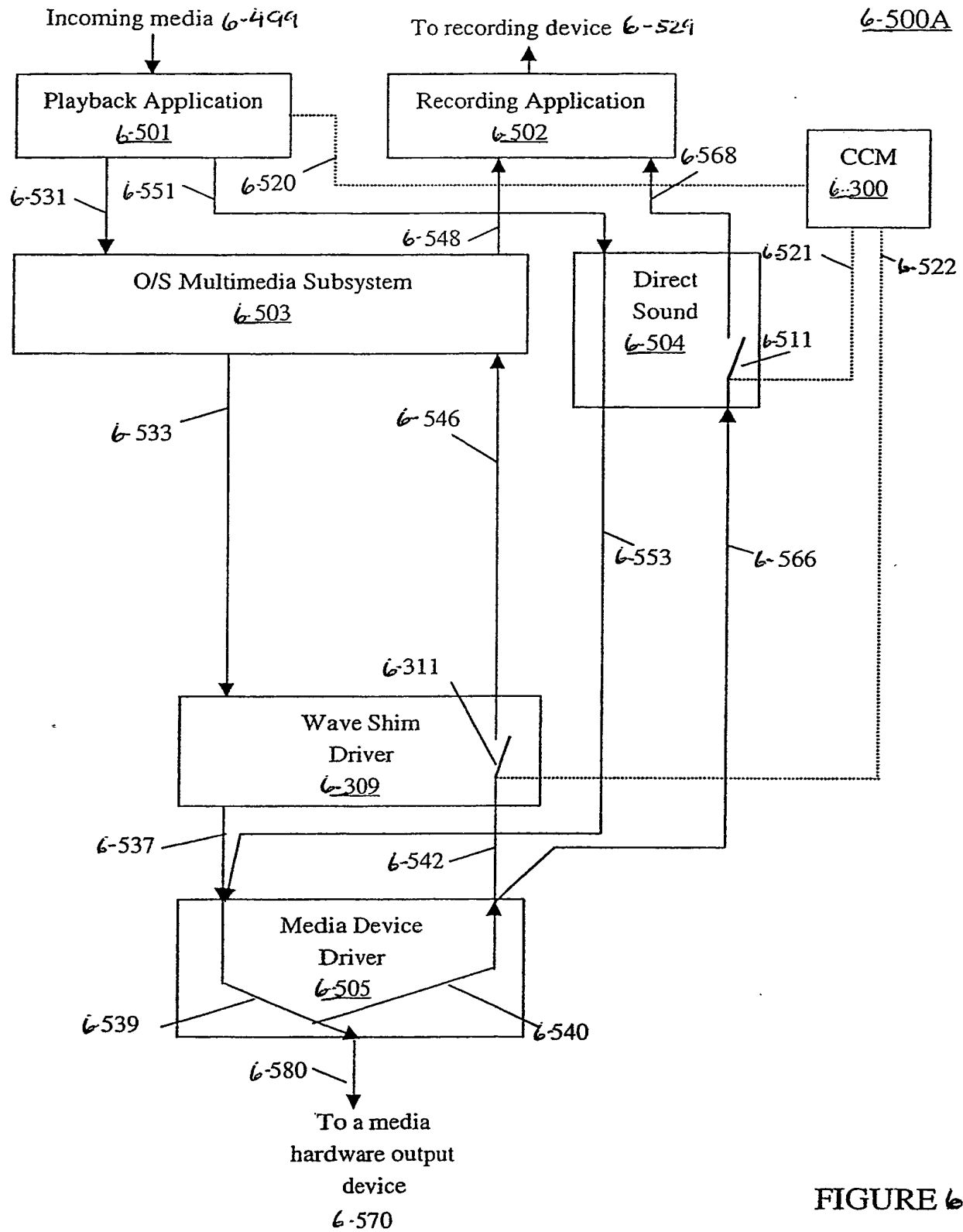


FIGURE 6-4



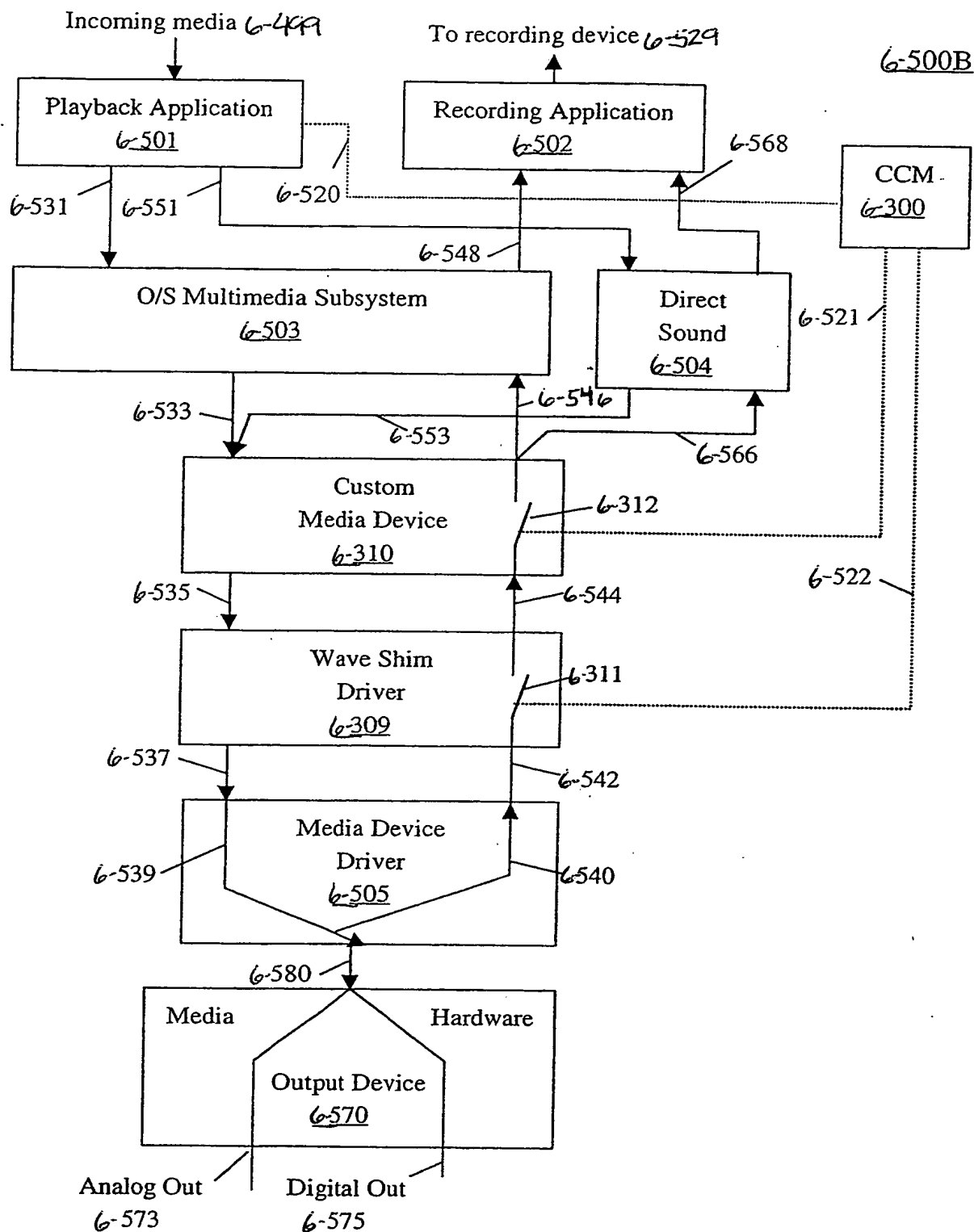


FIGURE 6-5B

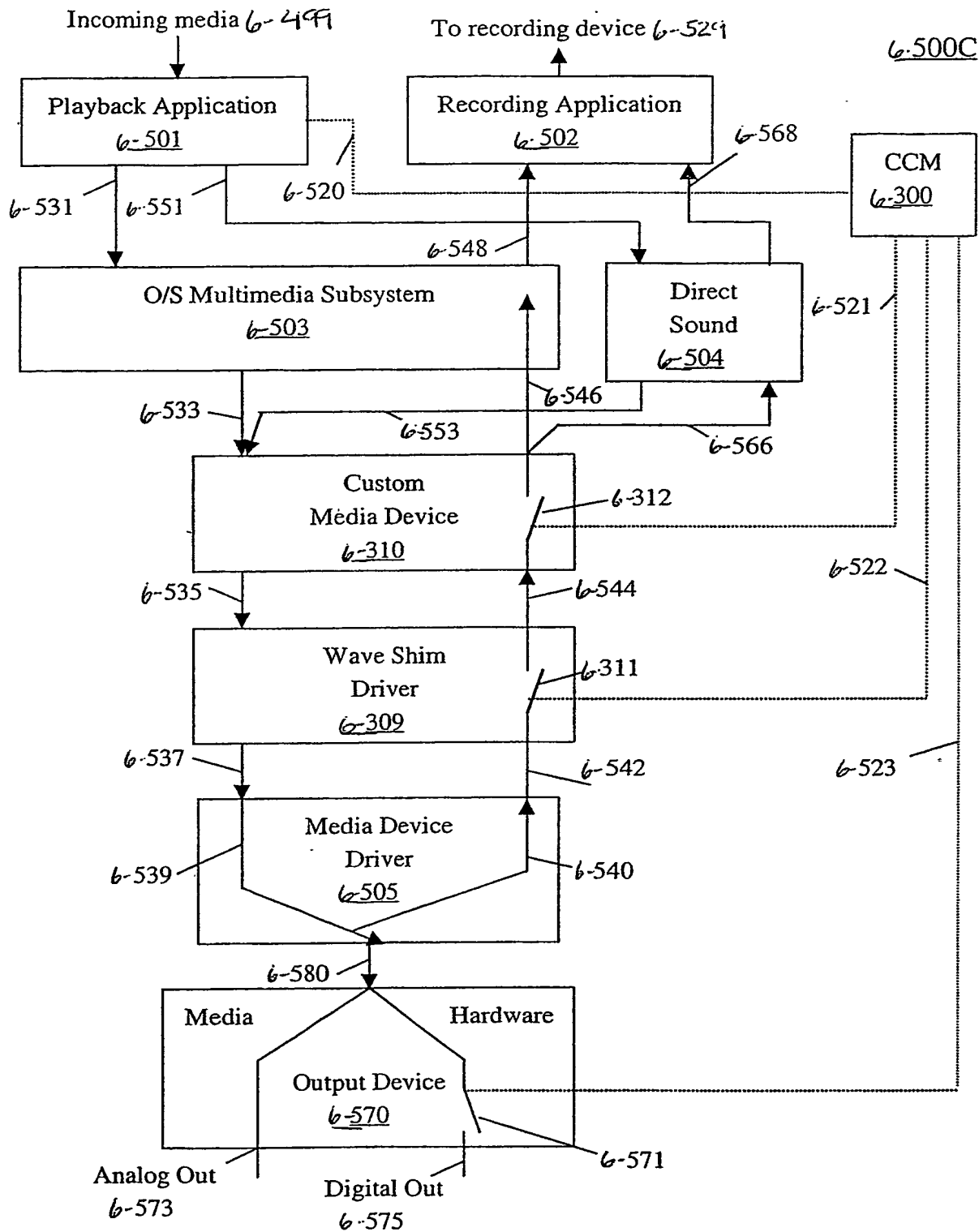


FIGURE 6-5C

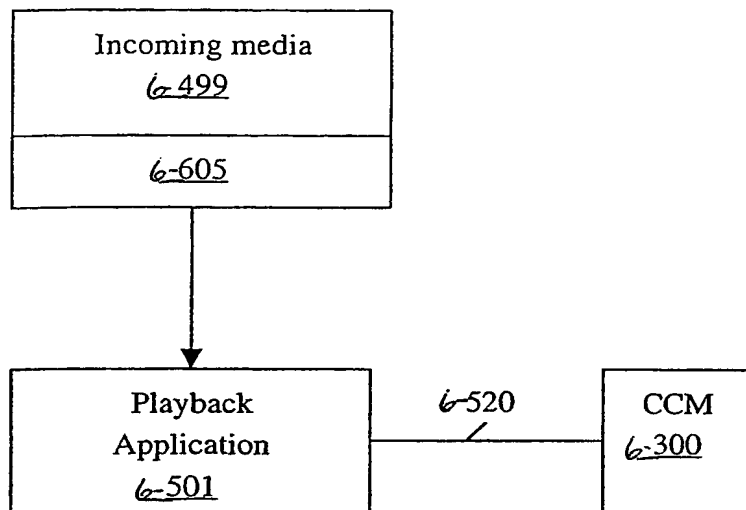
6-600

FIGURE 6-6A

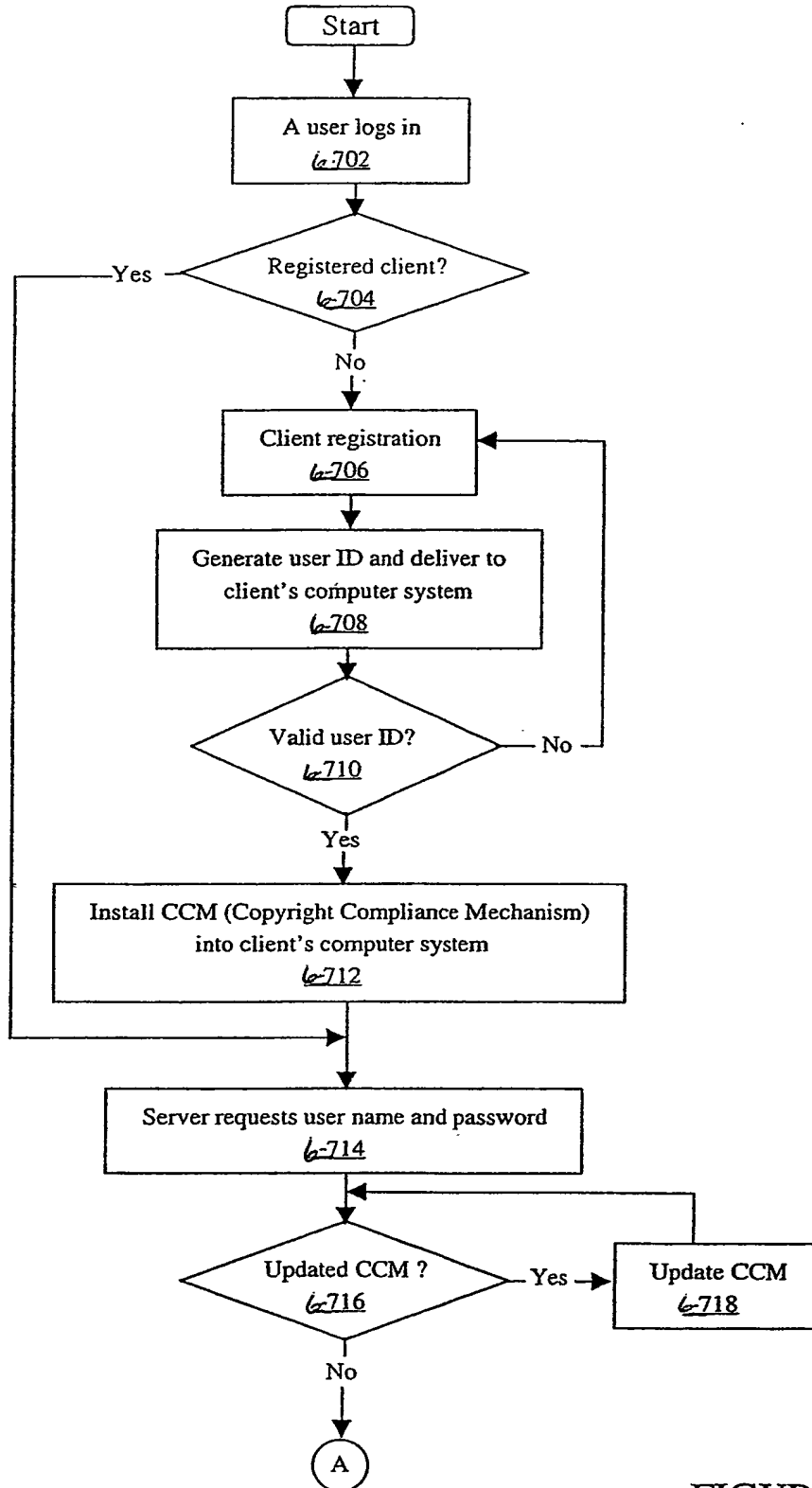
6-700

FIGURE 6-7A

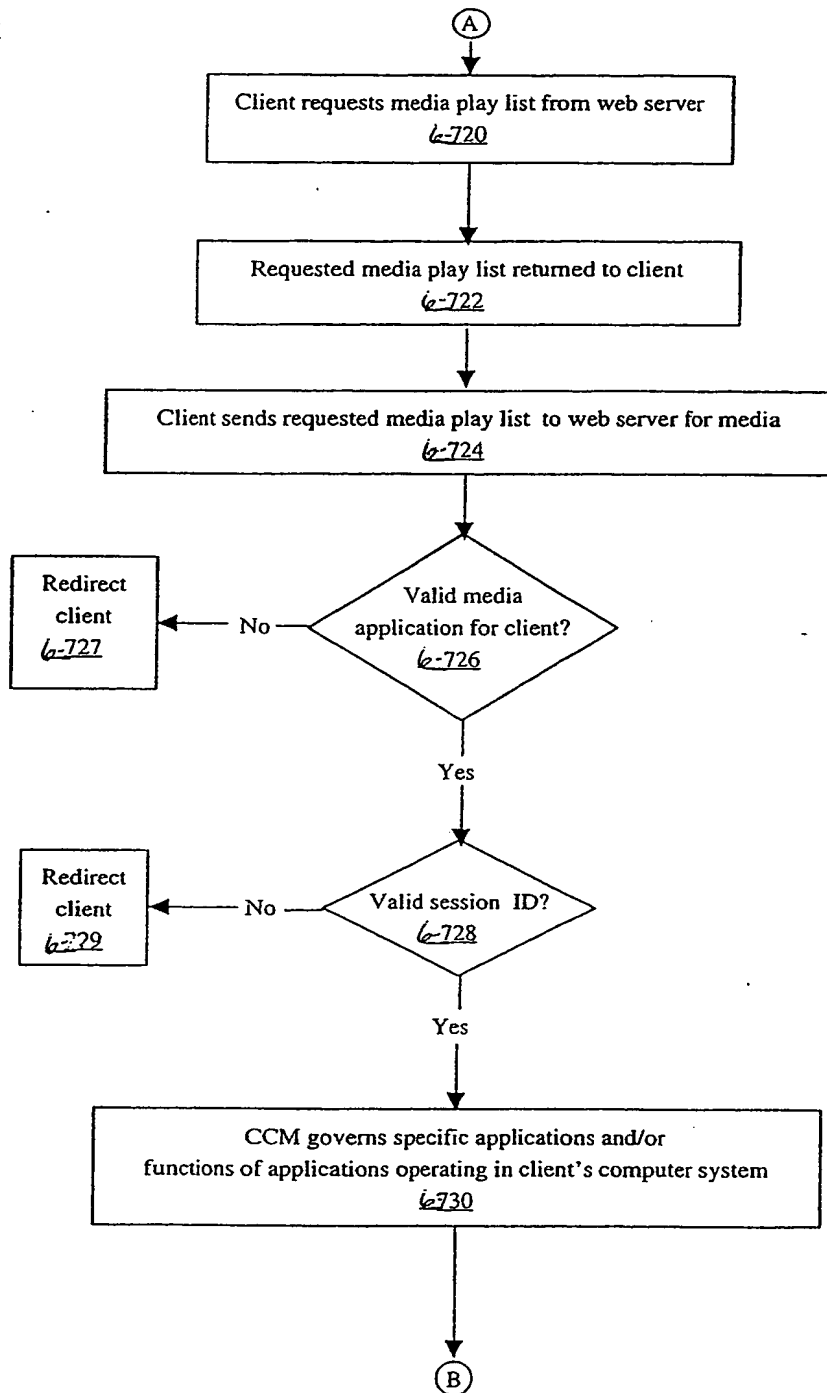
6-700

FIGURE 6-7B

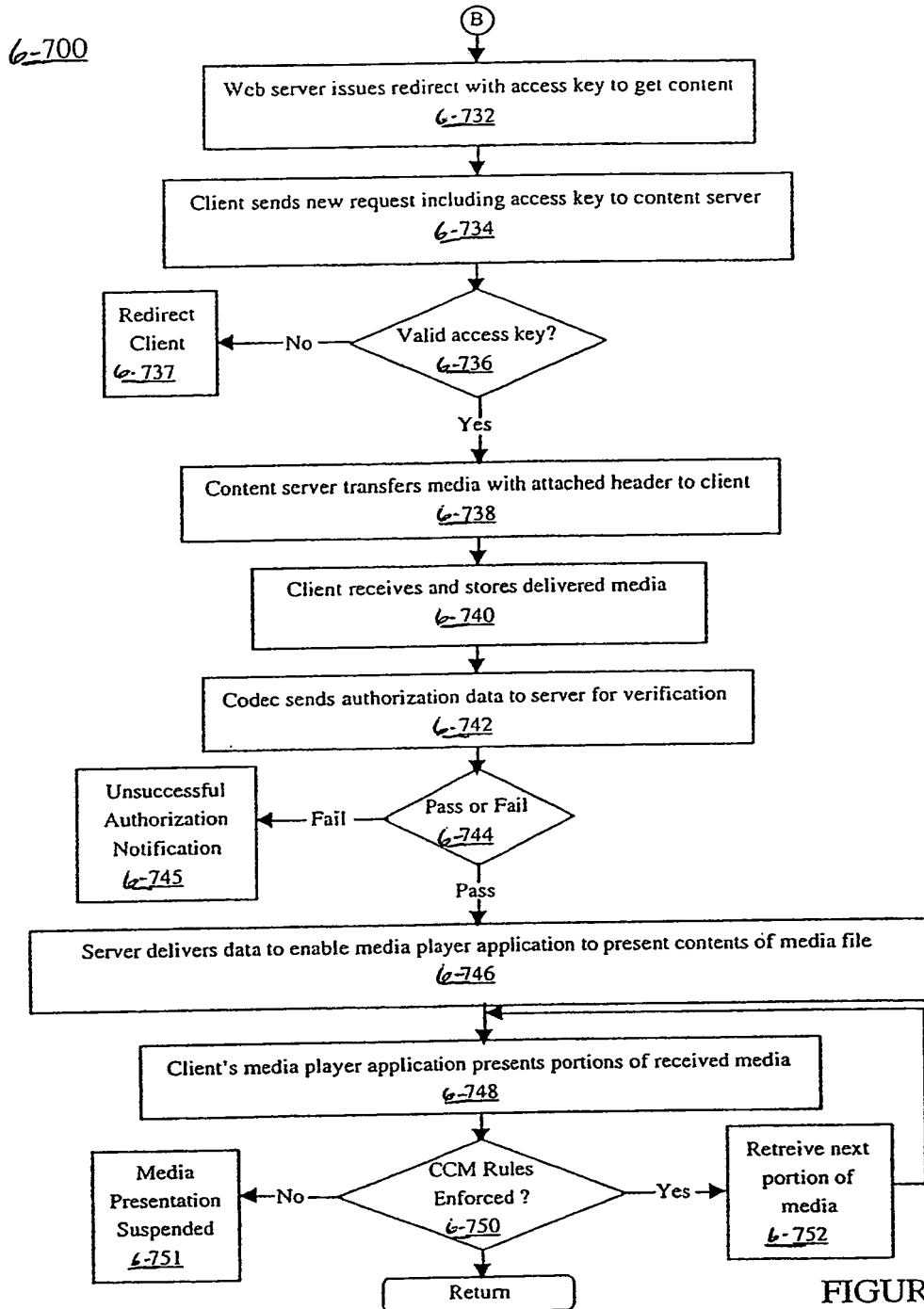


FIGURE 6-7C

6-800

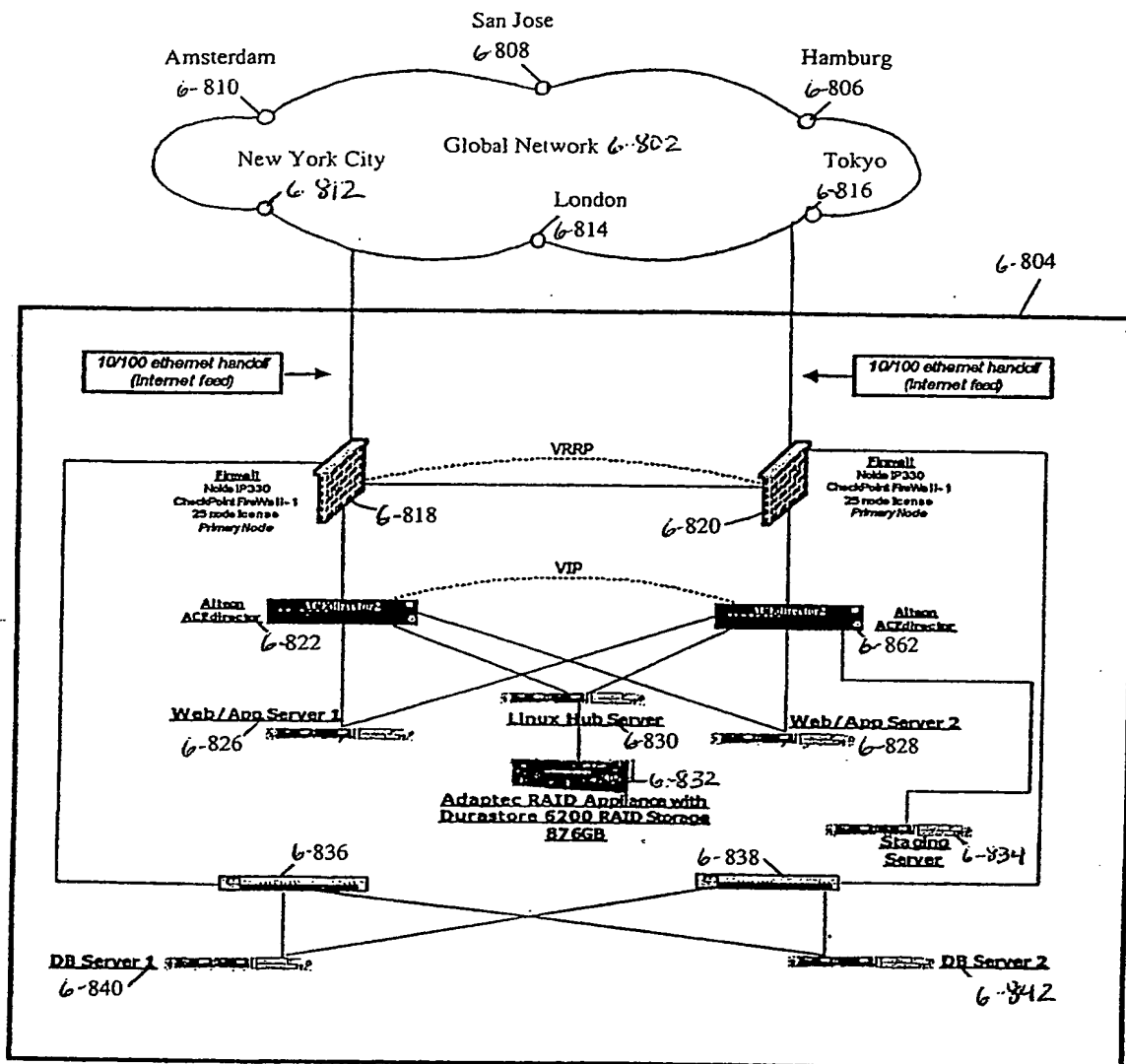


FIGURE 6-8

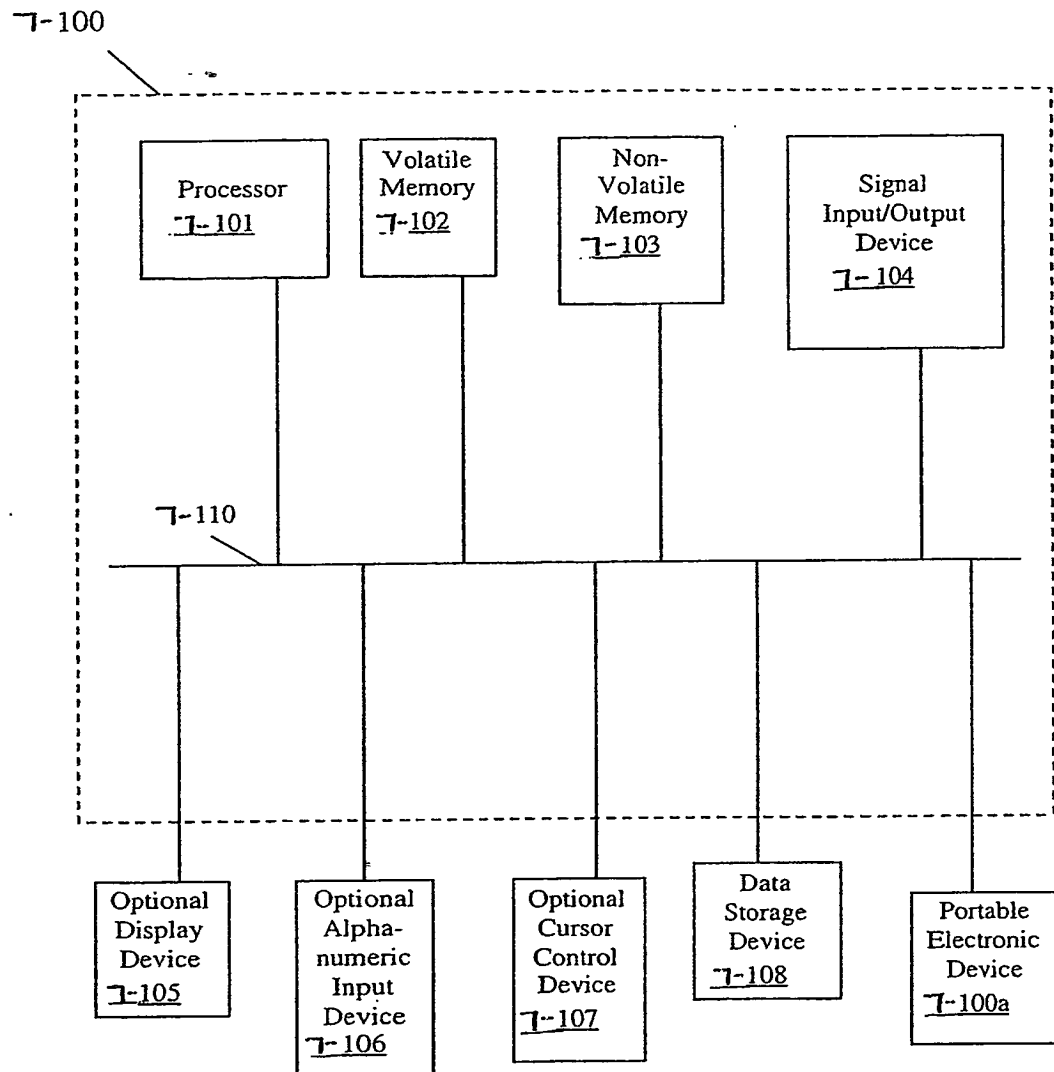


FIGURE 7-1

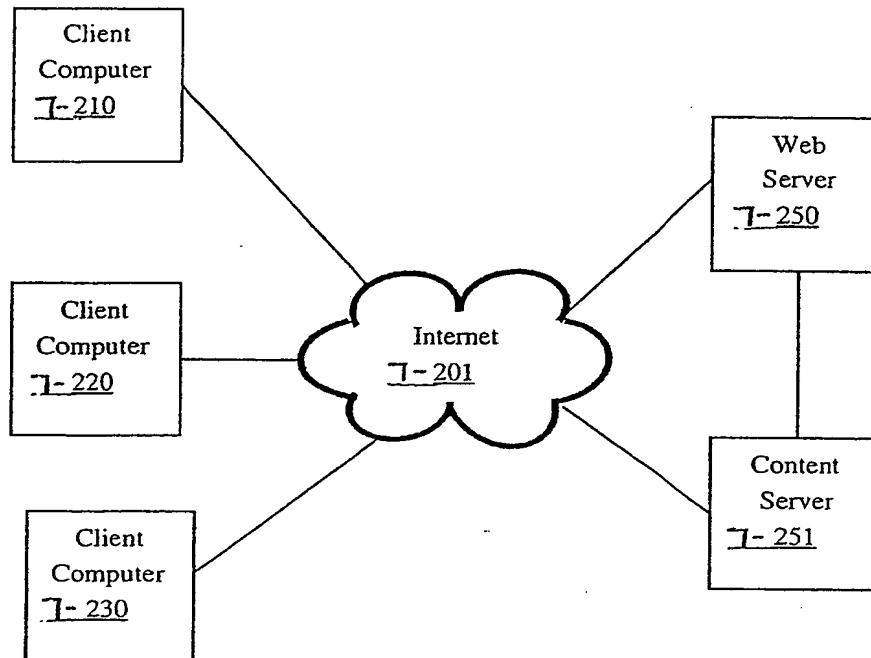
7-200

FIGURE 7-2

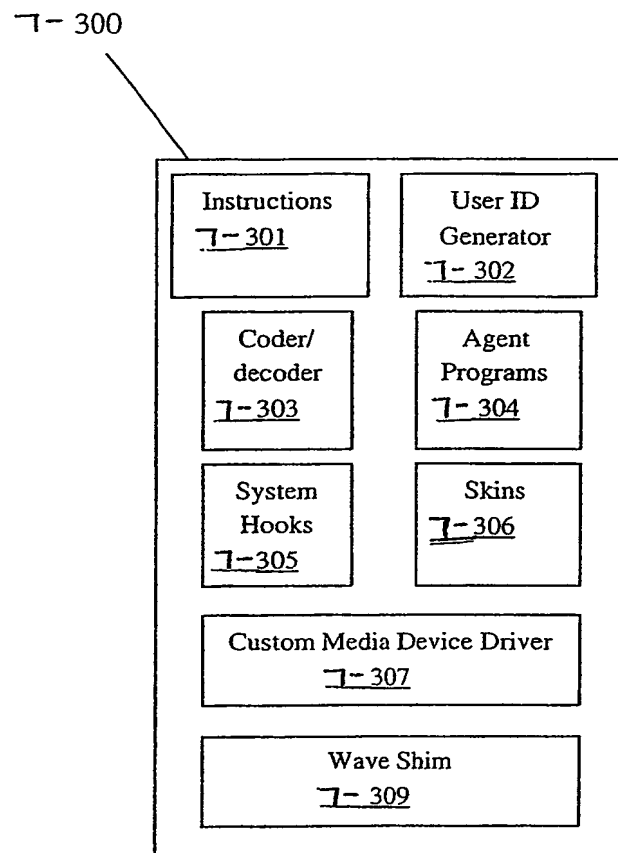


FIGURE 7-3

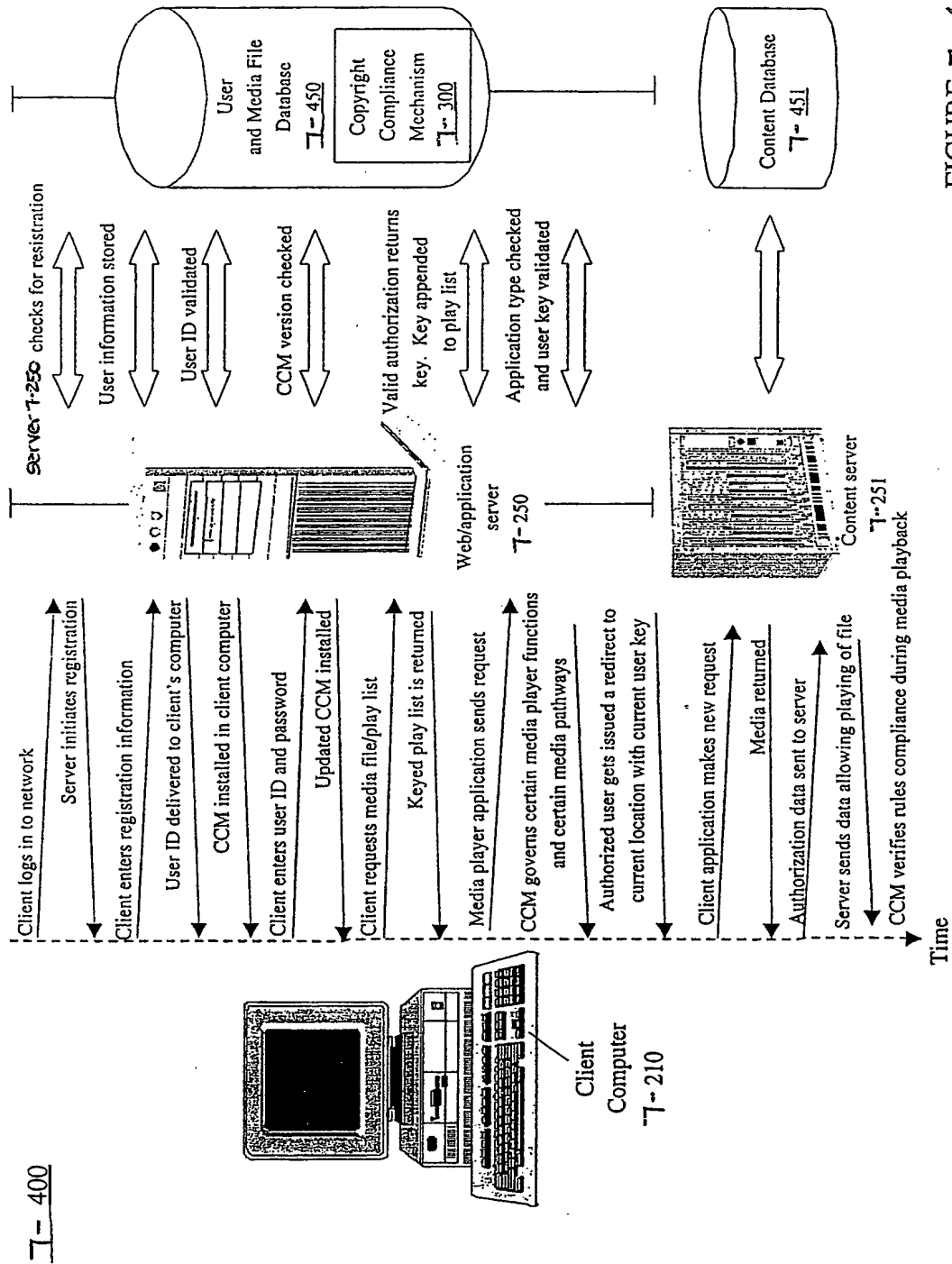


FIGURE 7-4

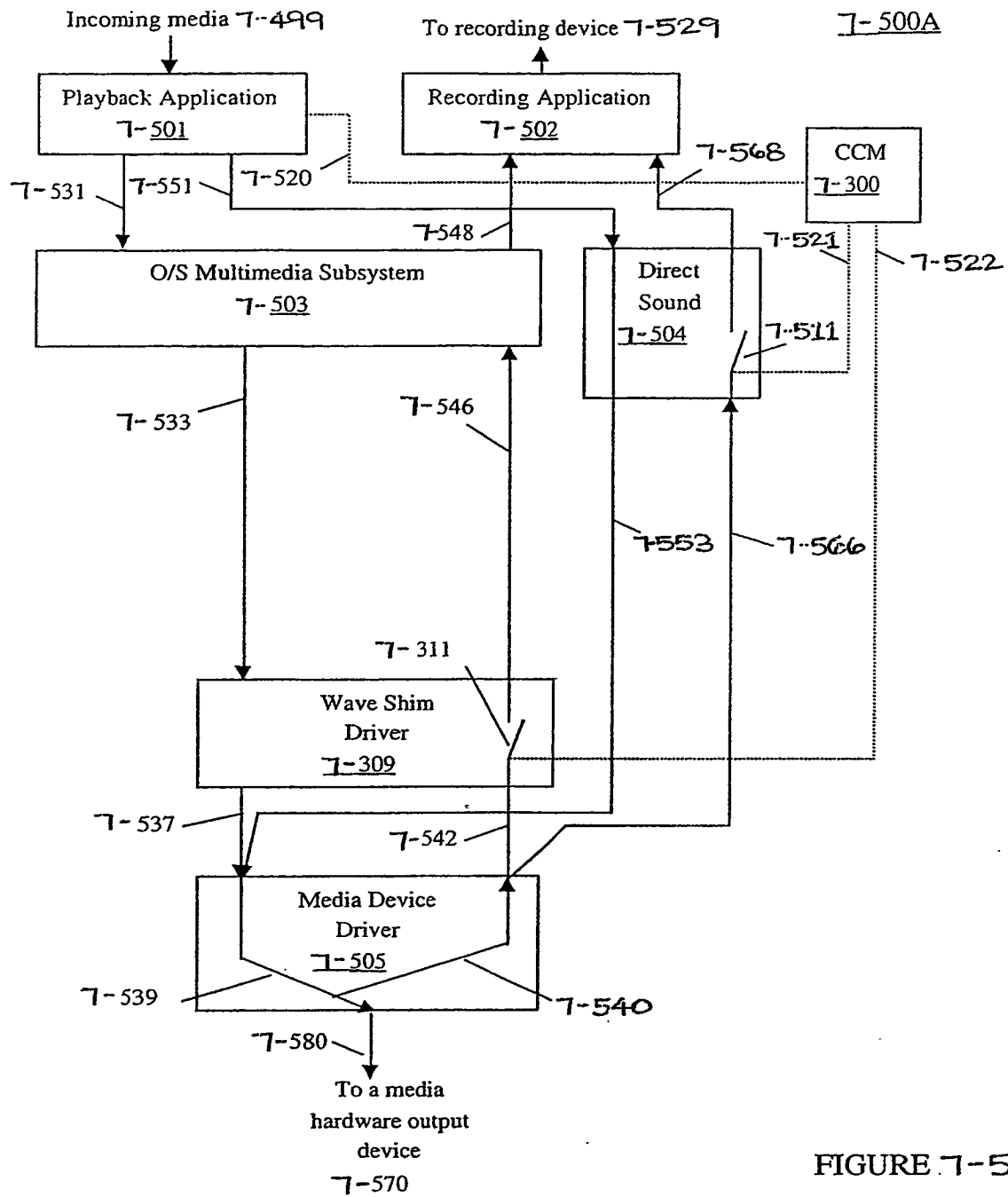


FIGURE 7-5A

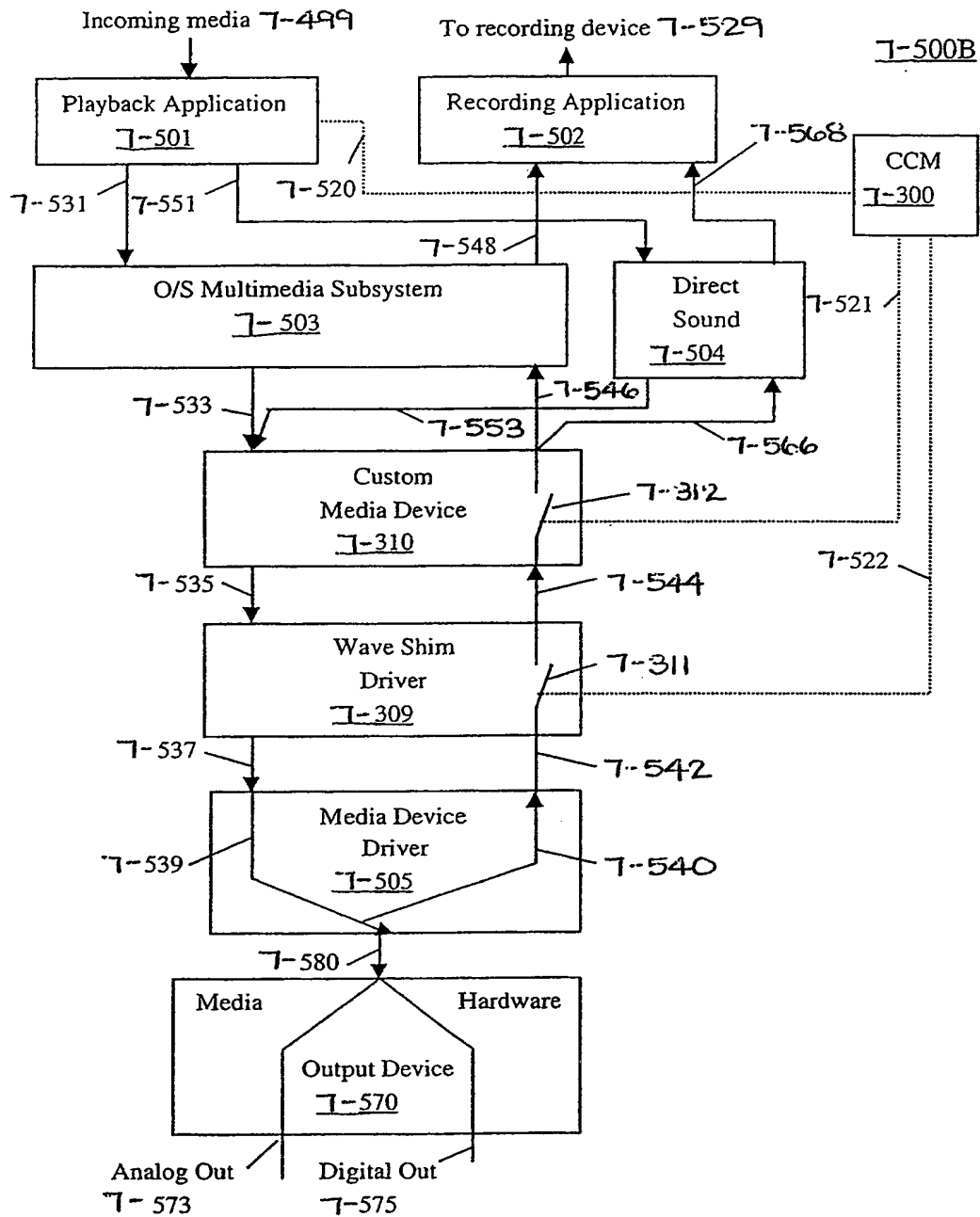


FIGURE 7-5B

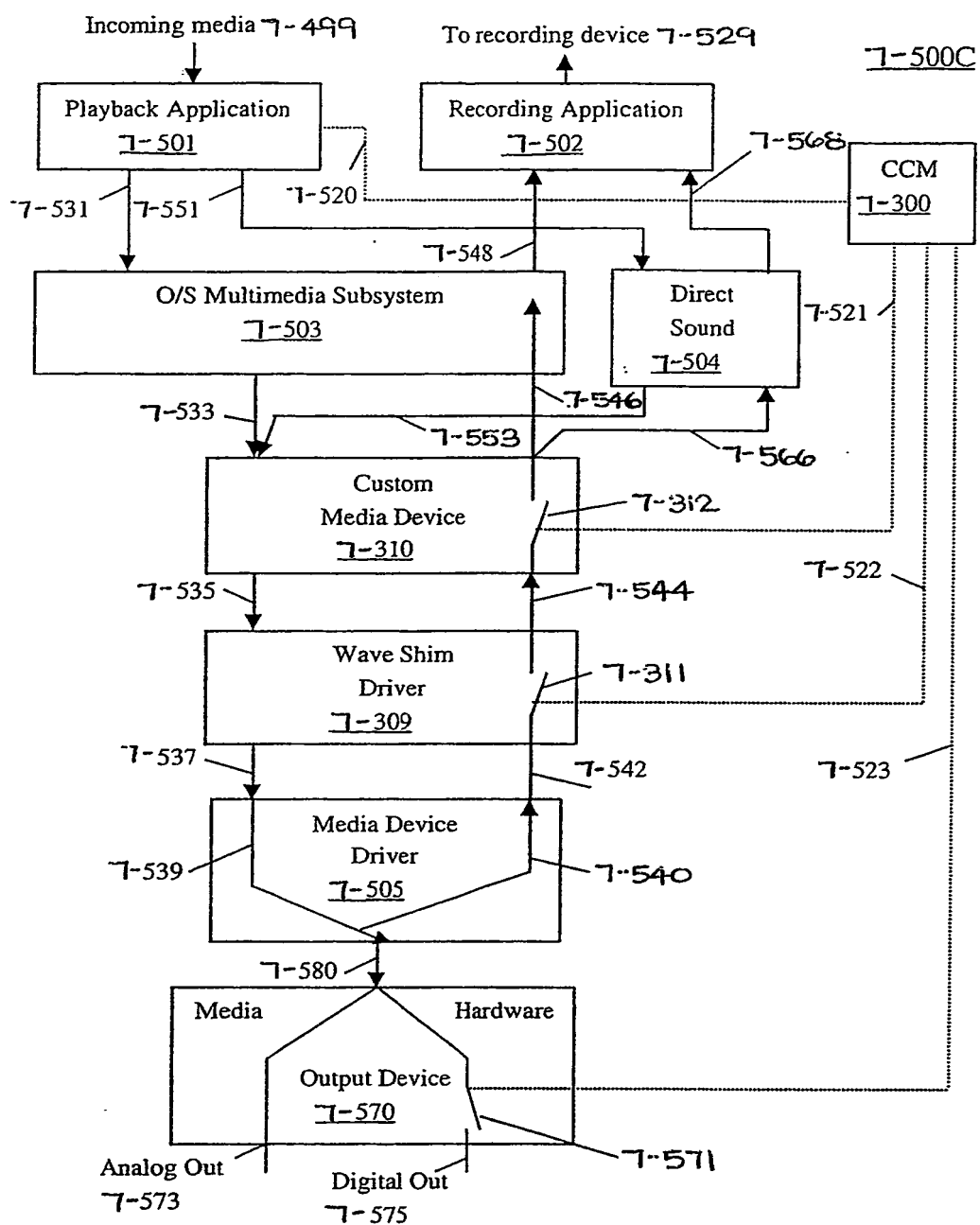


FIGURE 7-5C

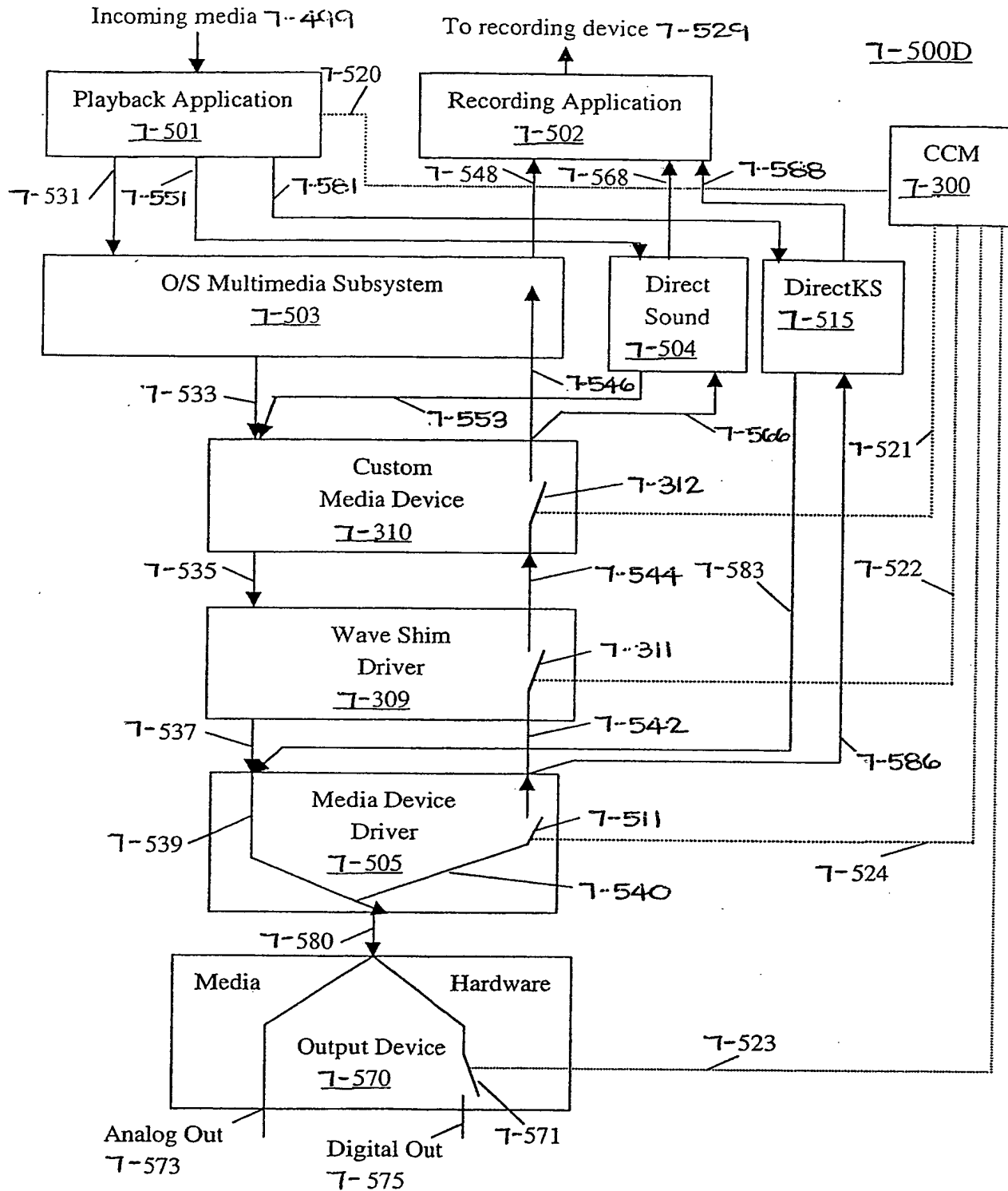


FIGURE 7-5D

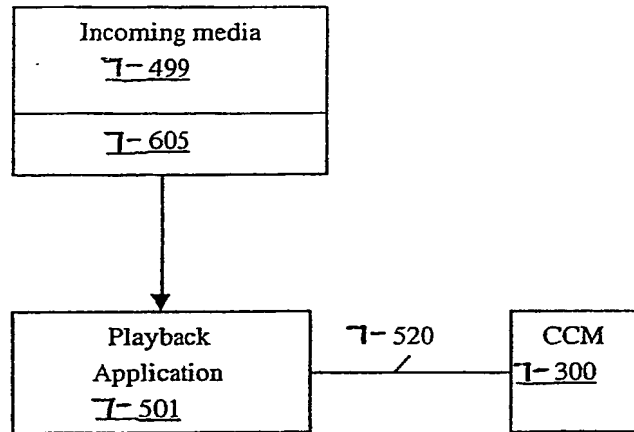
7-600

FIGURE 7-6A

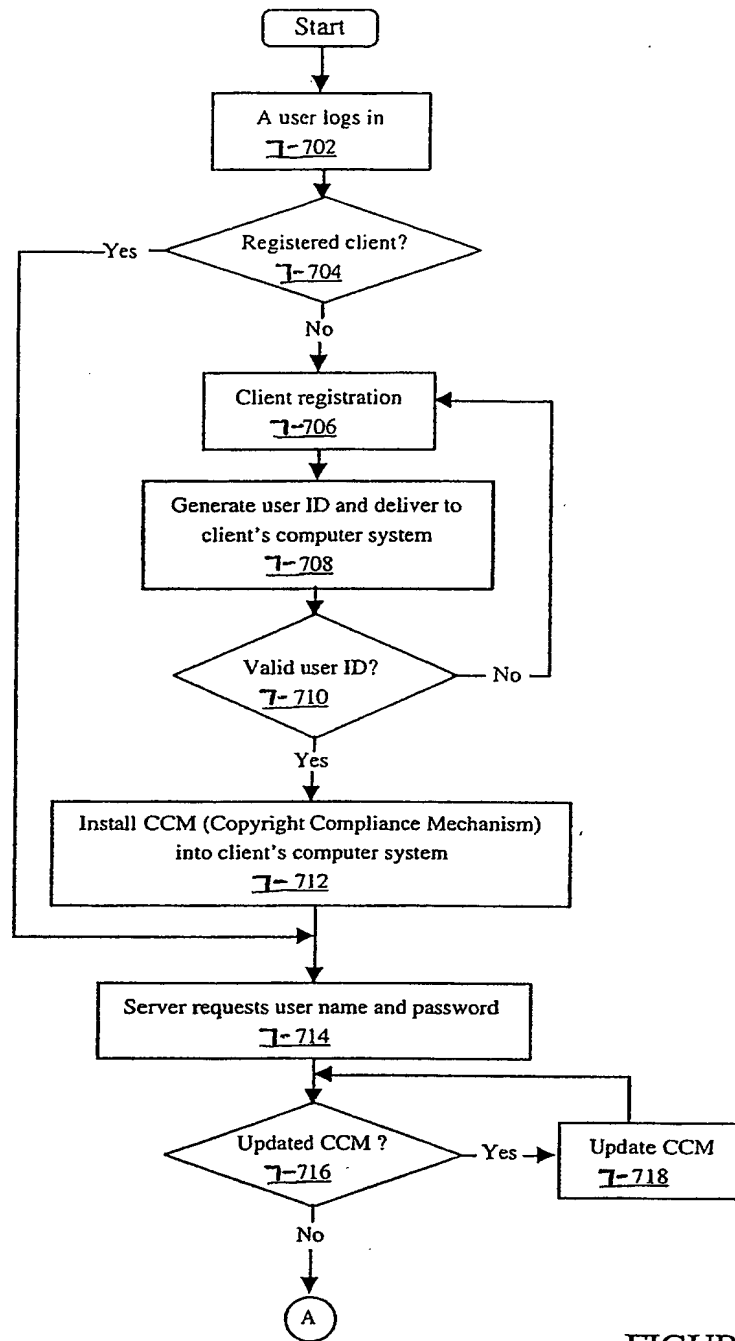
7-700

FIGURE 7-7A

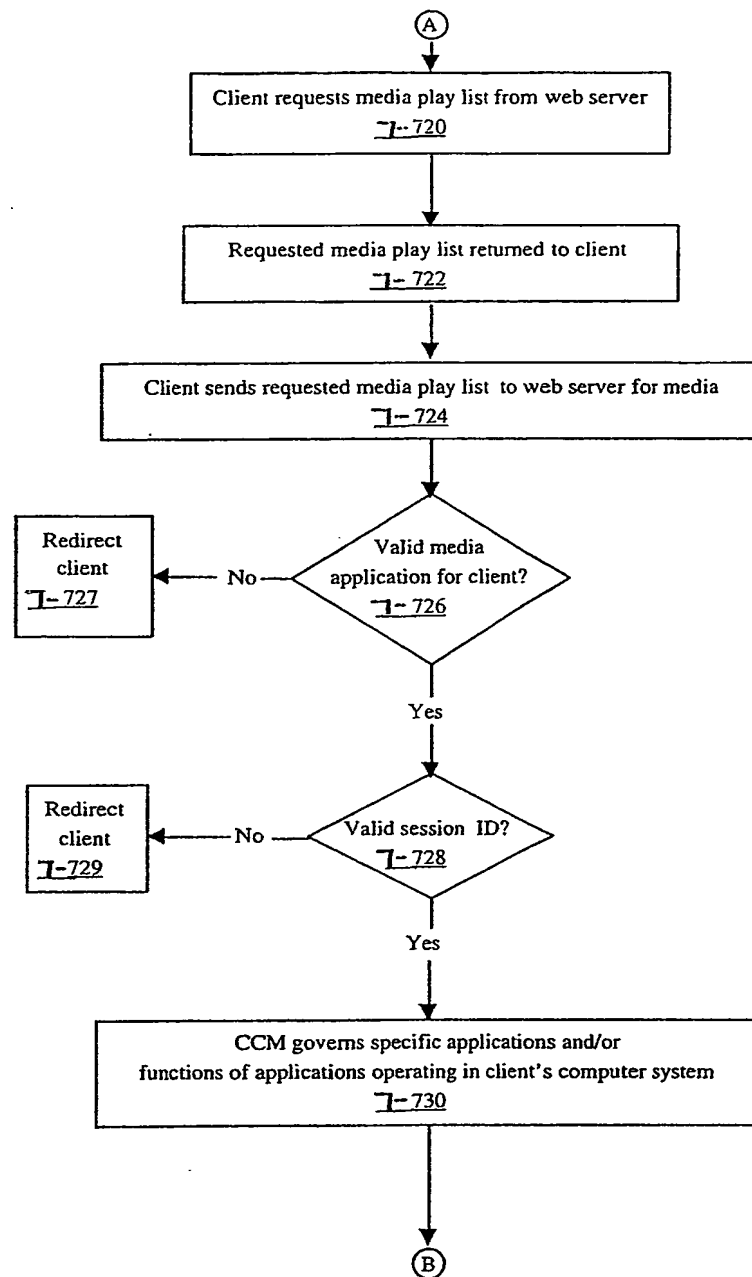
1-700

FIGURE 1-1B

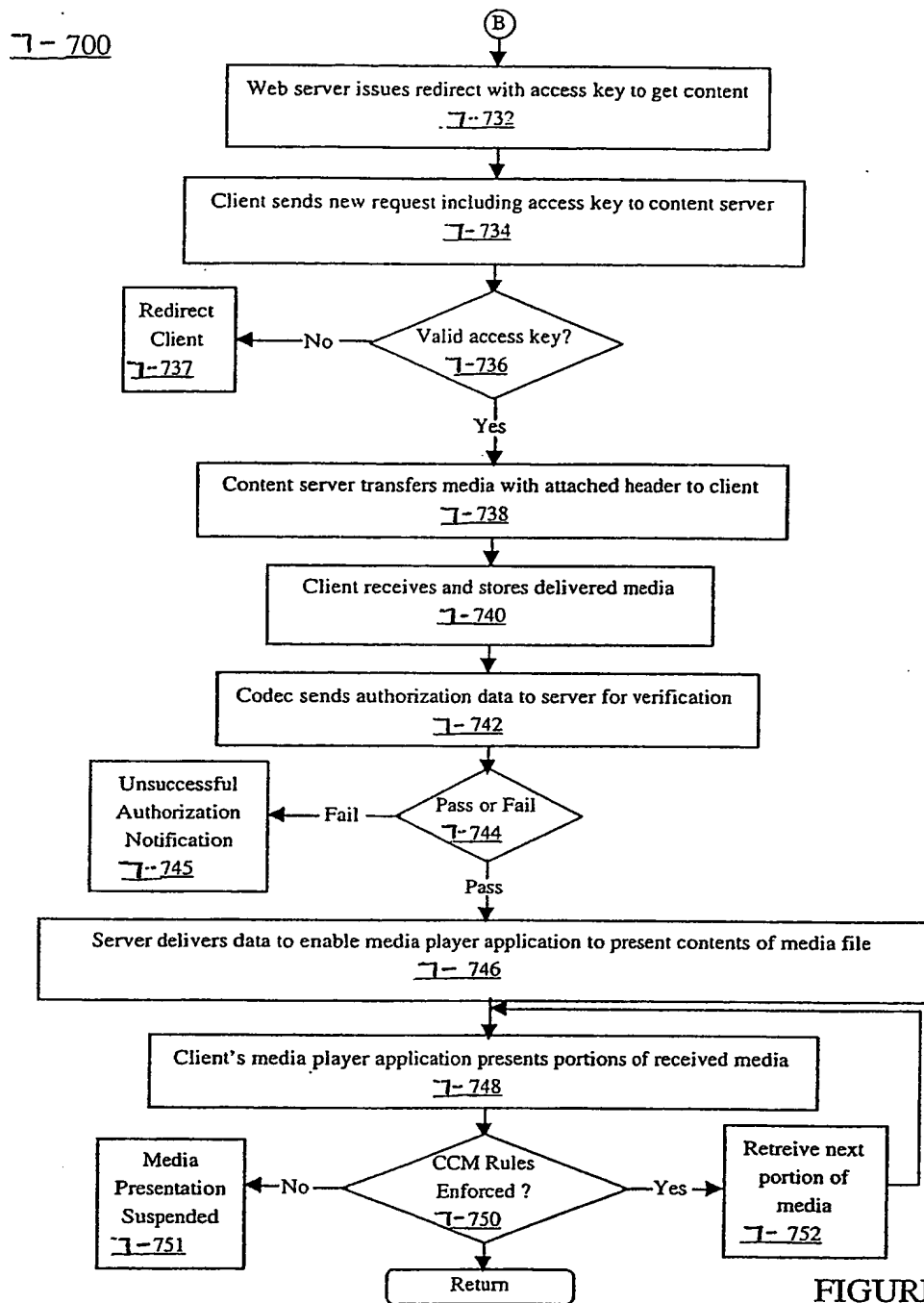


FIGURE 7-7C

7-800

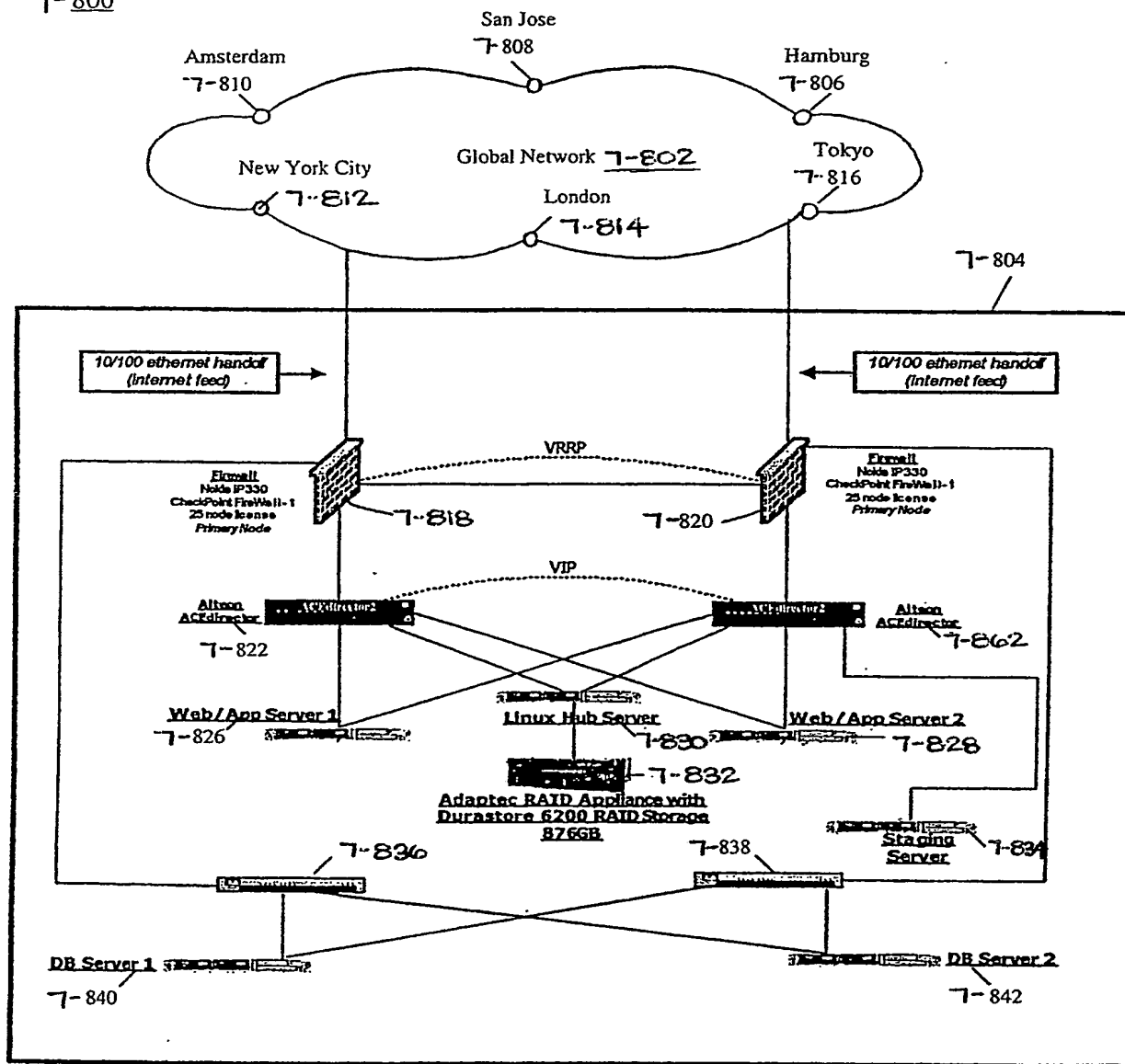


FIGURE 7-8

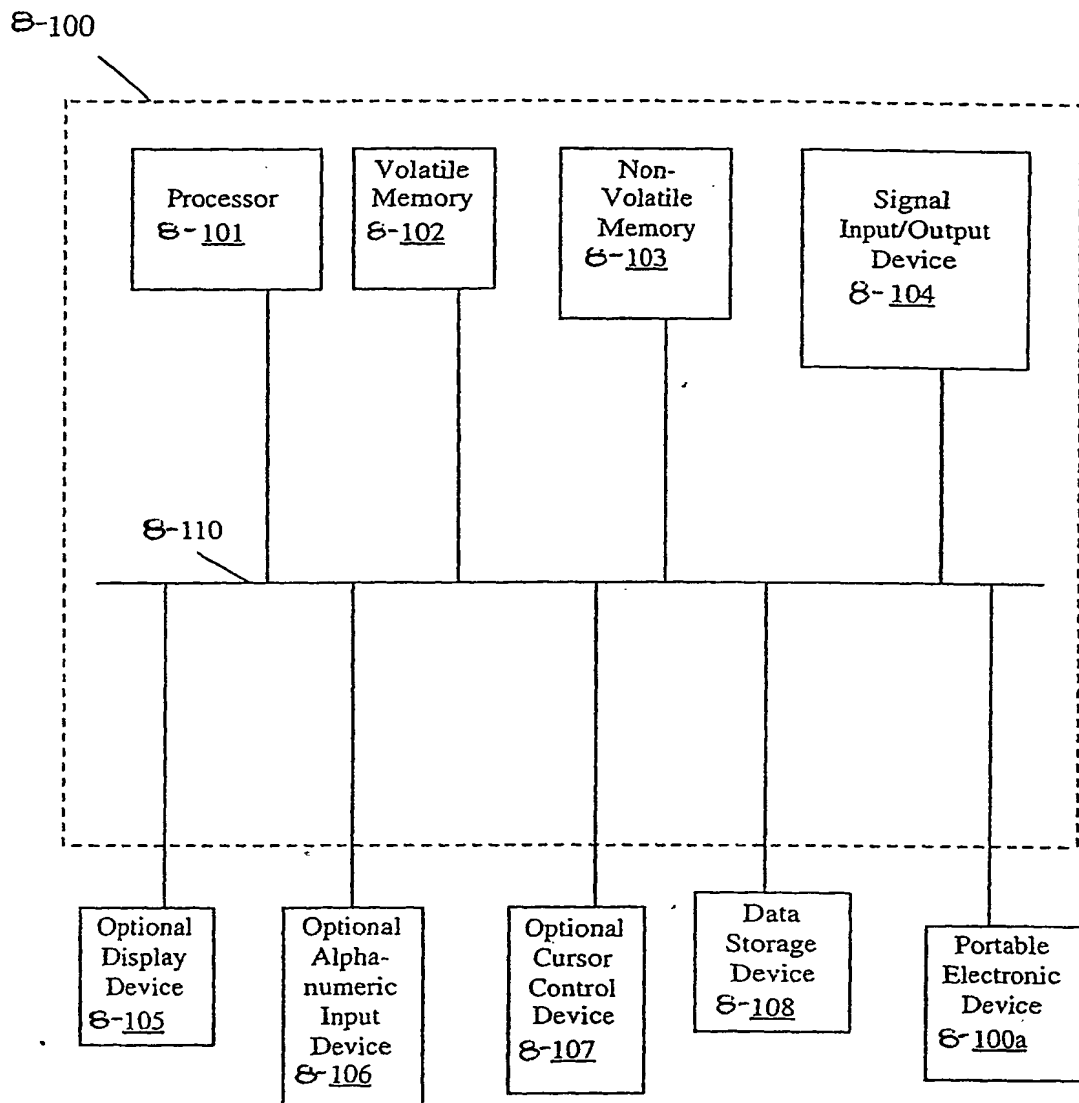


FIGURE 8-1

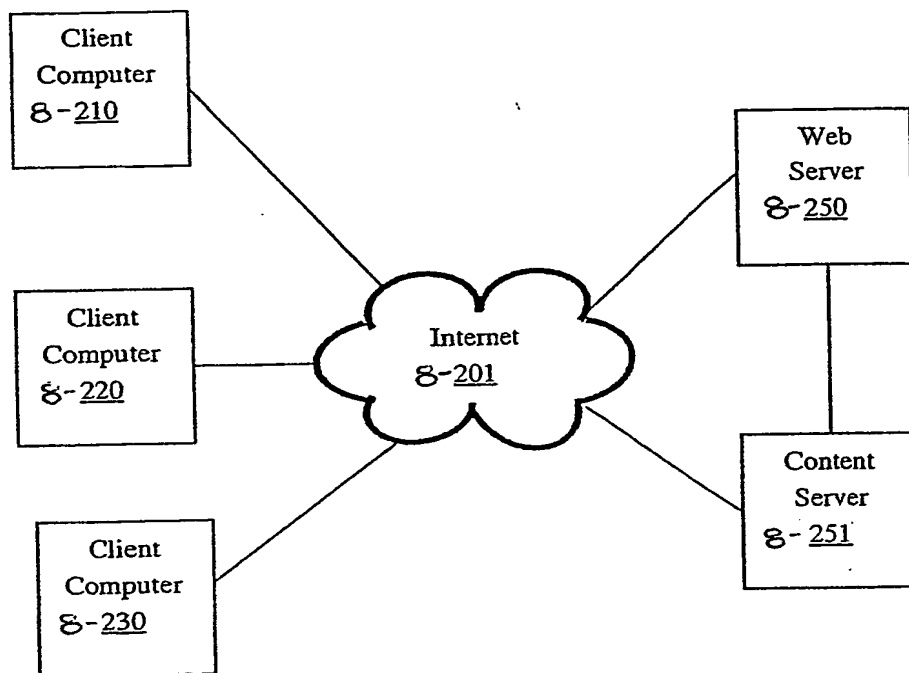
8-200

FIGURE 8-2

8-300

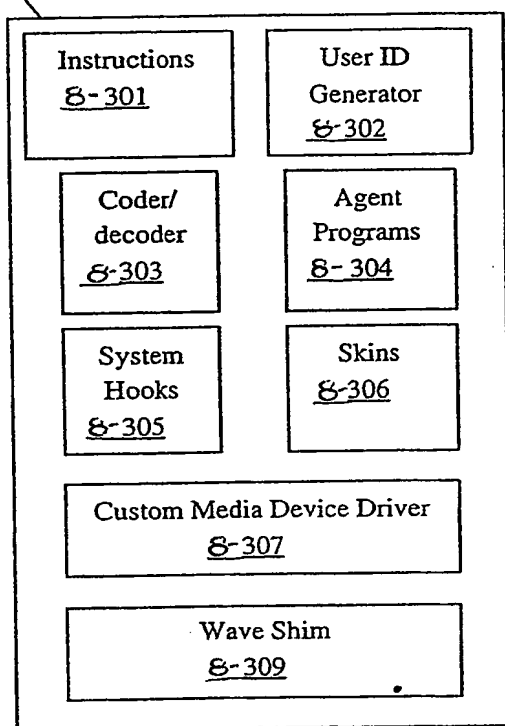


FIGURE 8-3

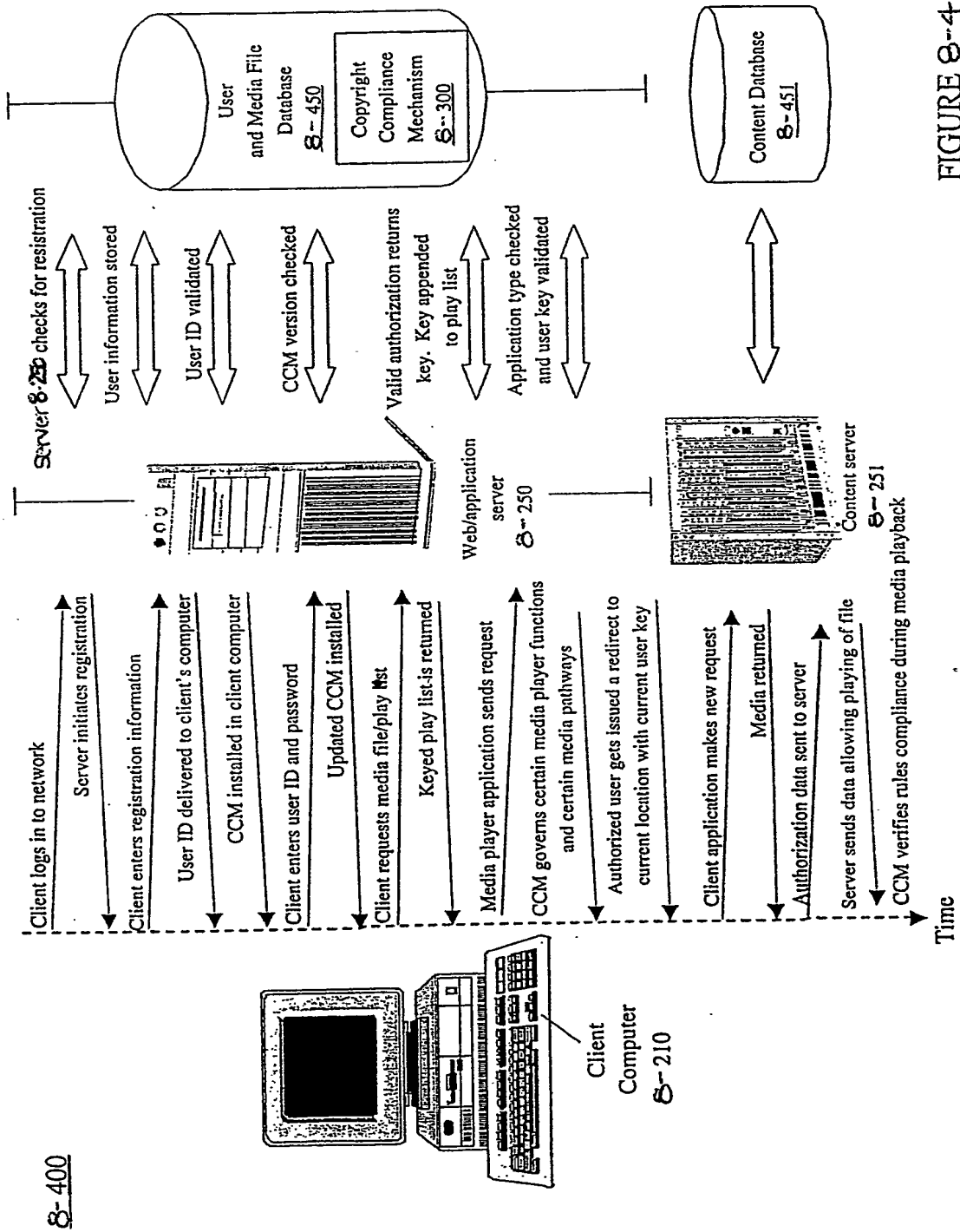


FIGURE 8-4

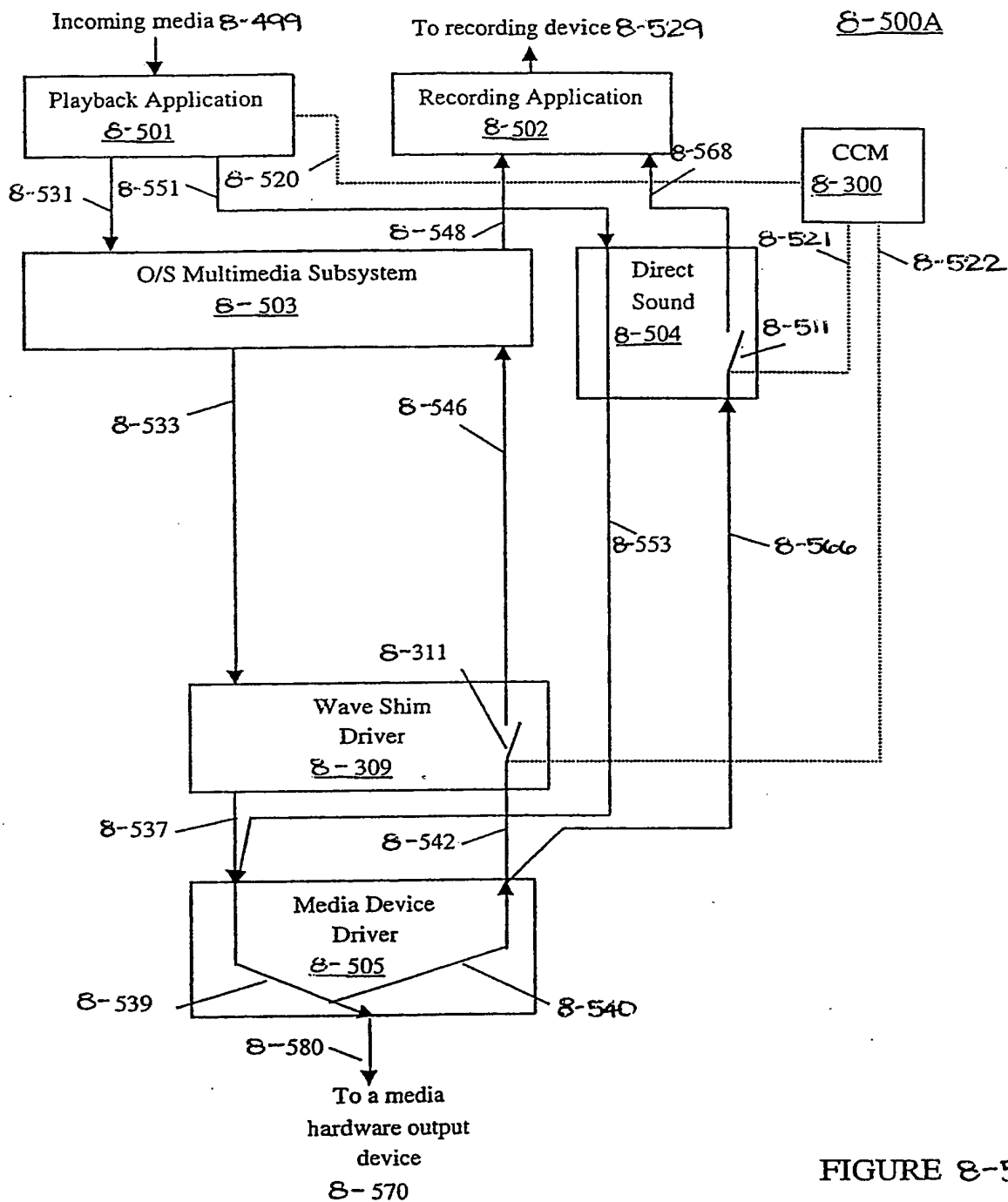


FIGURE 8-5A

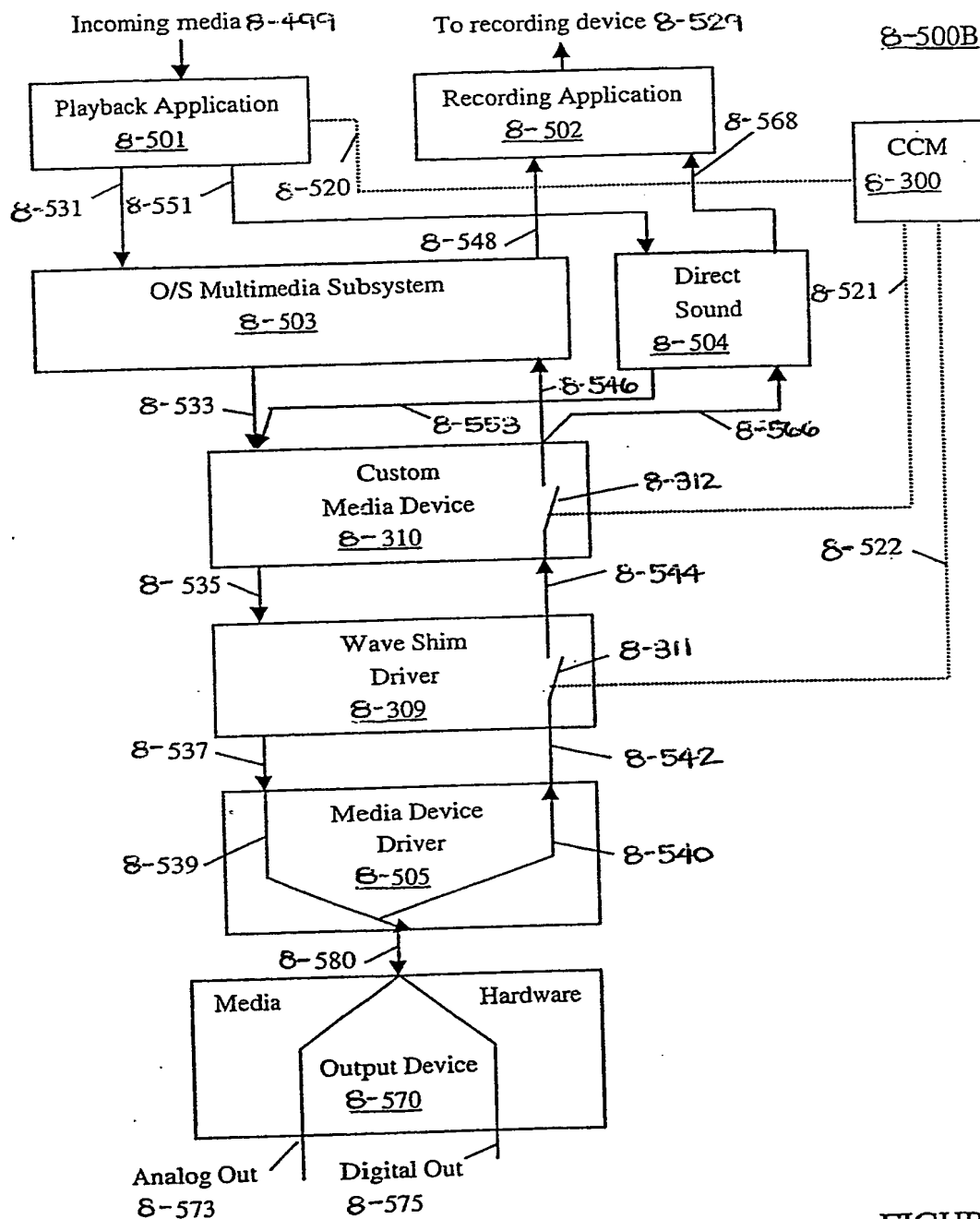


FIGURE 8-5B

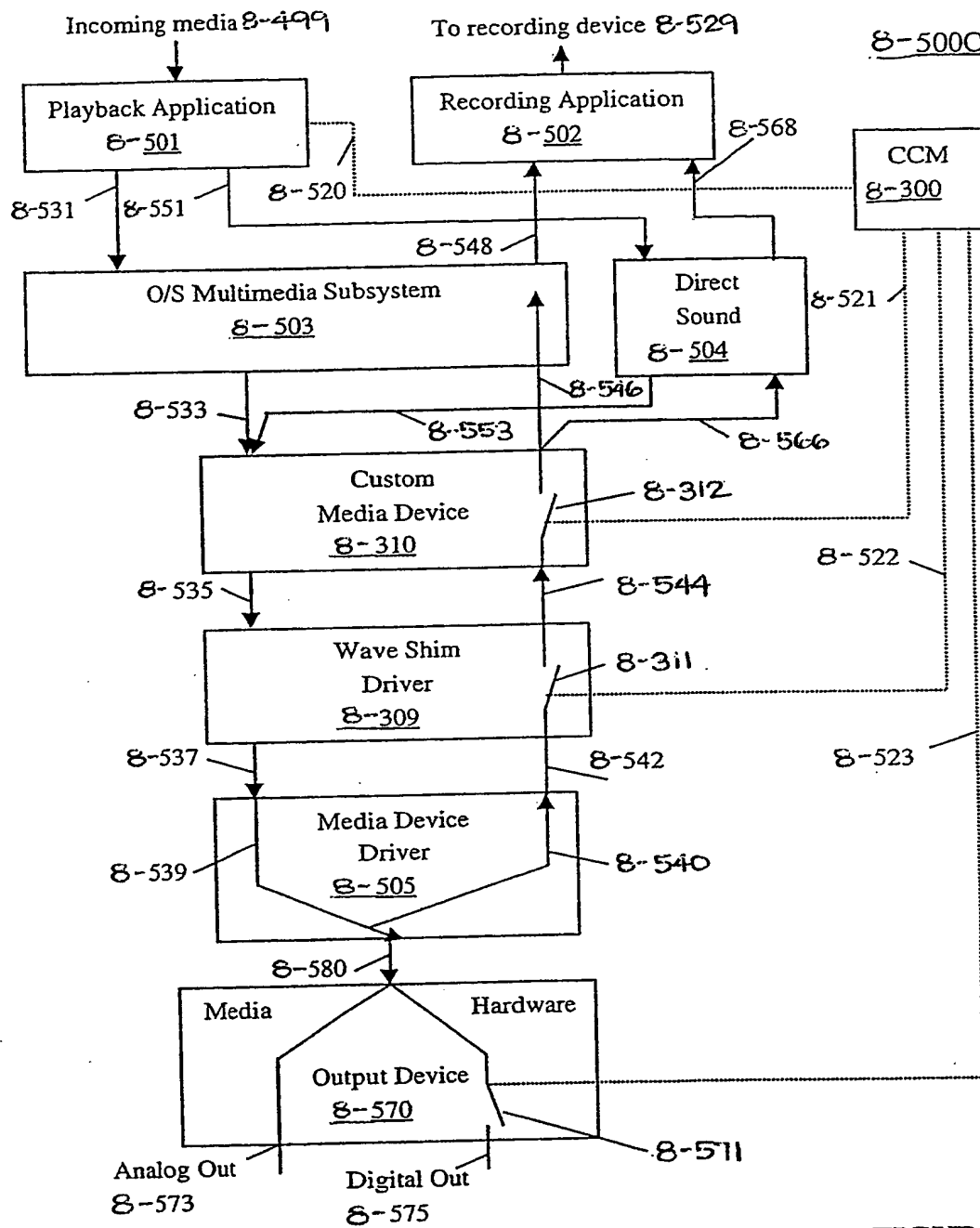


FIGURE 8-5C

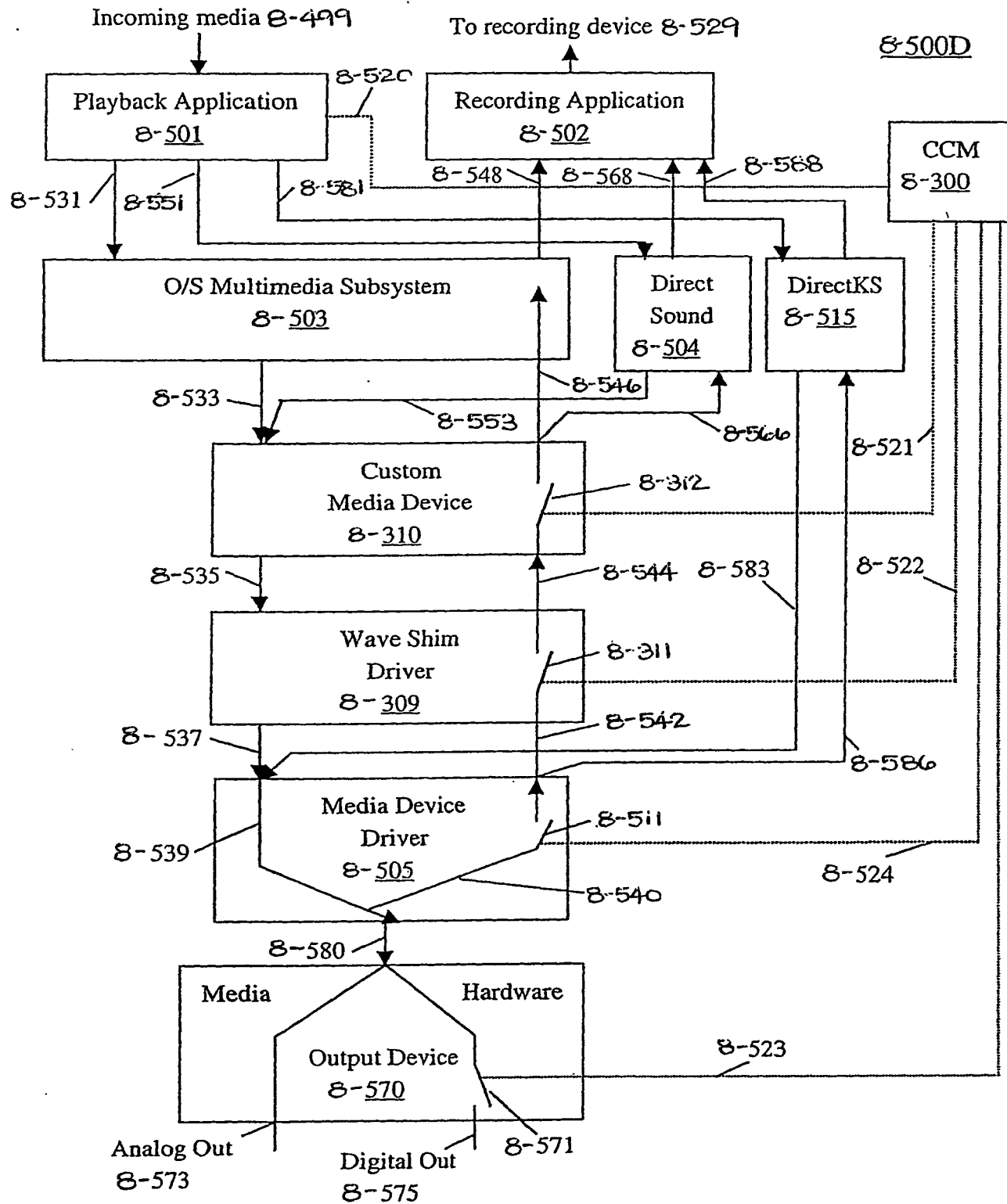


FIGURE 8-5D

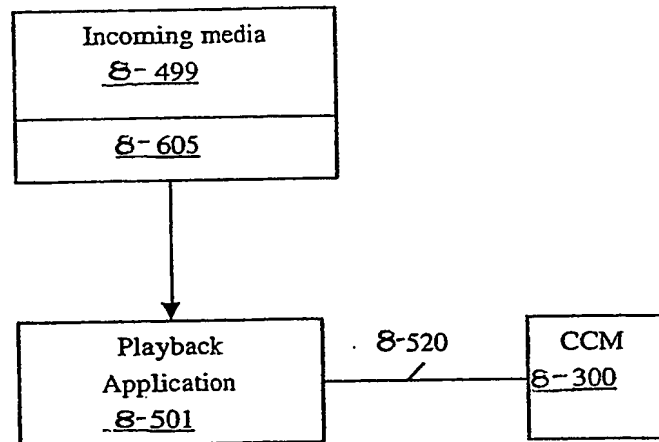
8- 600

FIGURE 8-6A

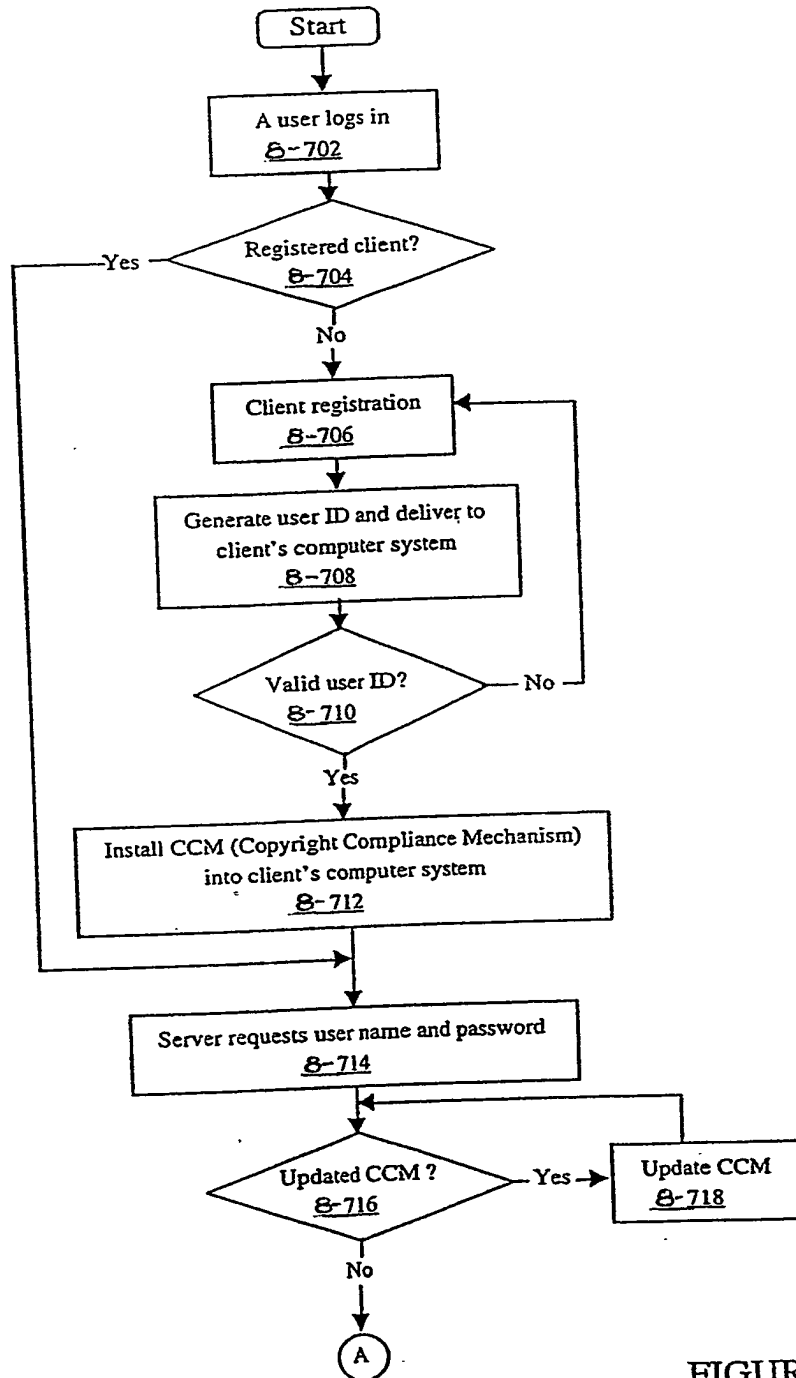
8-700

FIGURE 8-1A

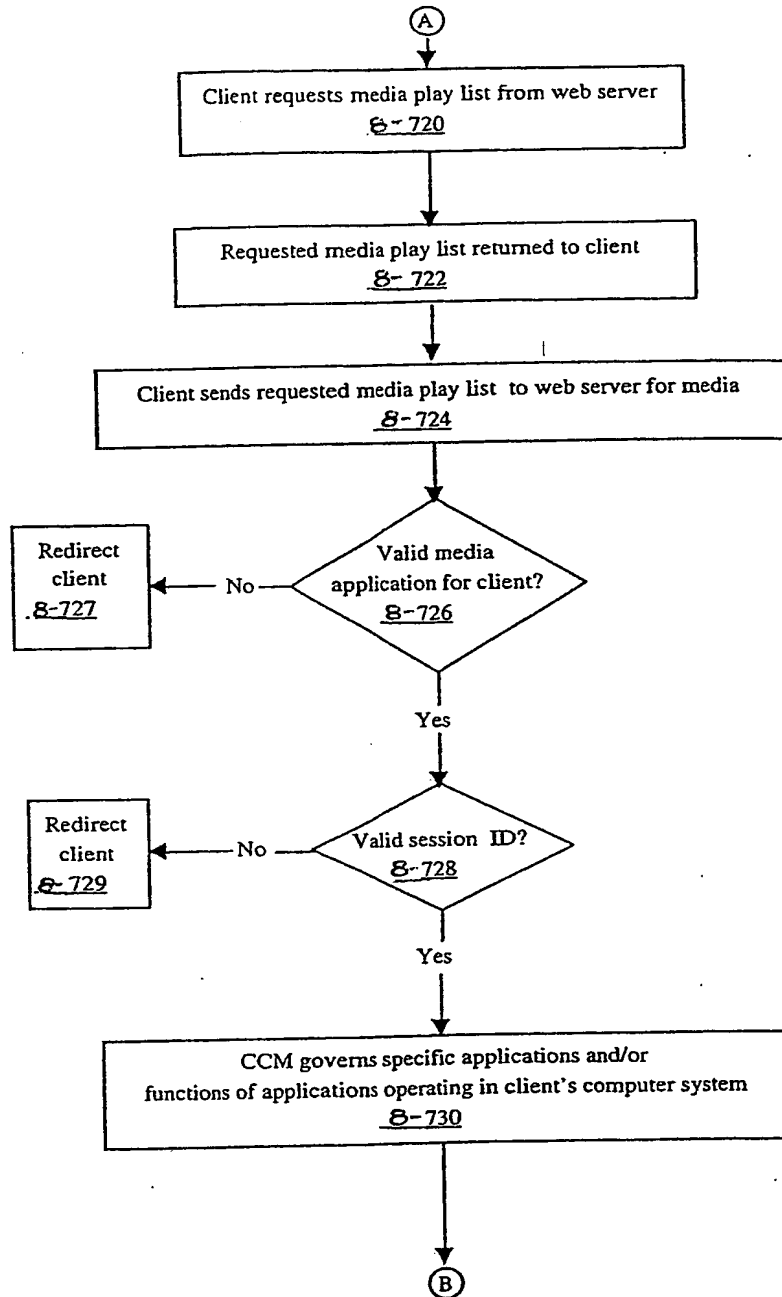
8-700

FIGURE 8-7B

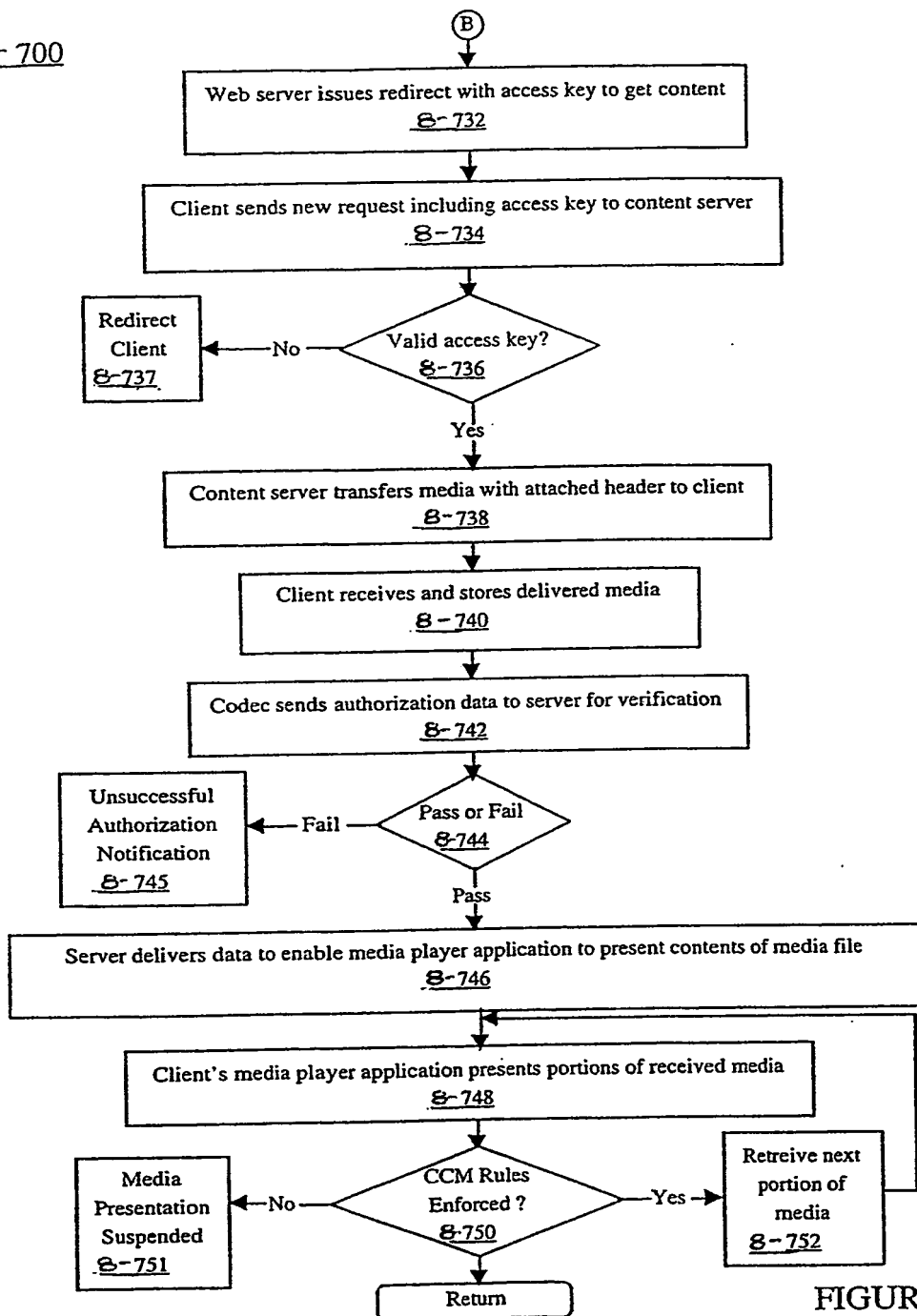
8-700

FIGURE 8-7C

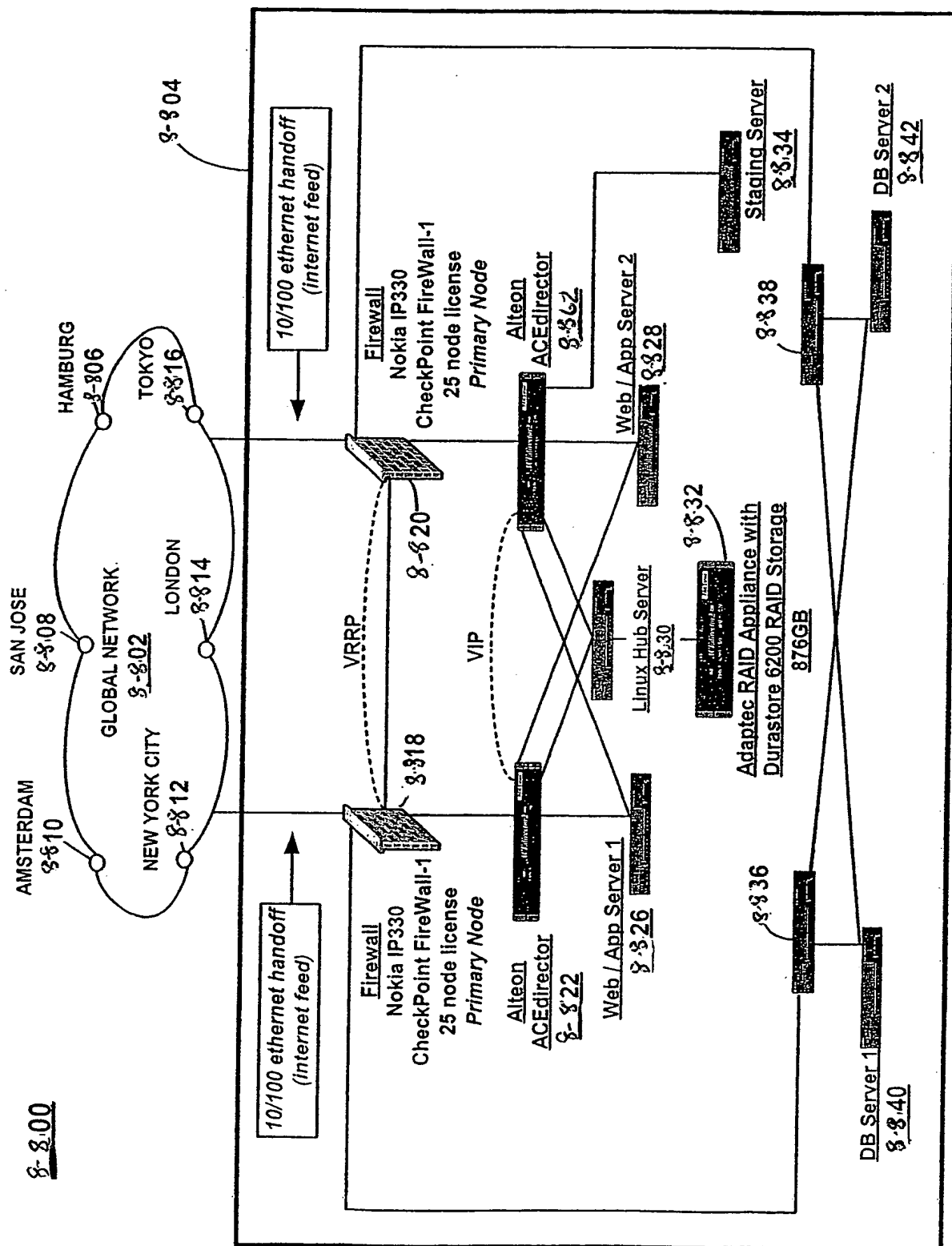


FIG. 8-8

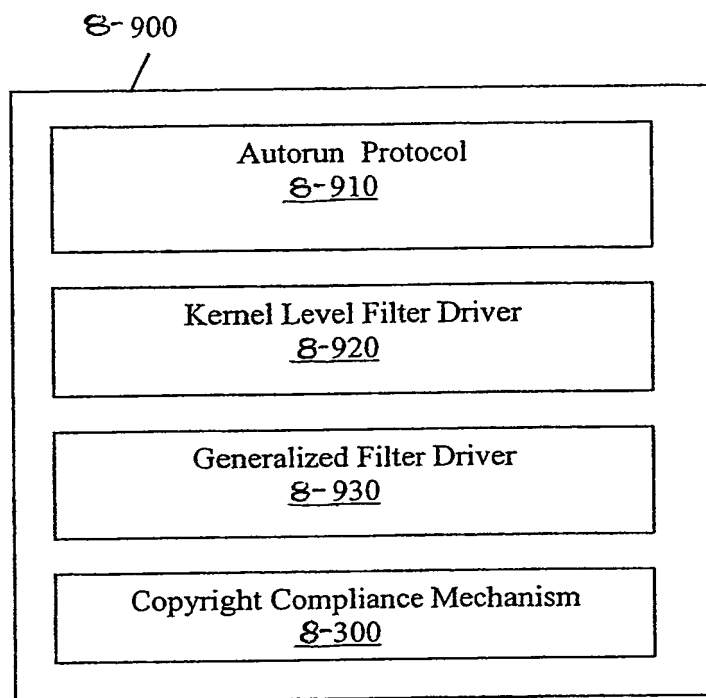


FIGURE 8-9

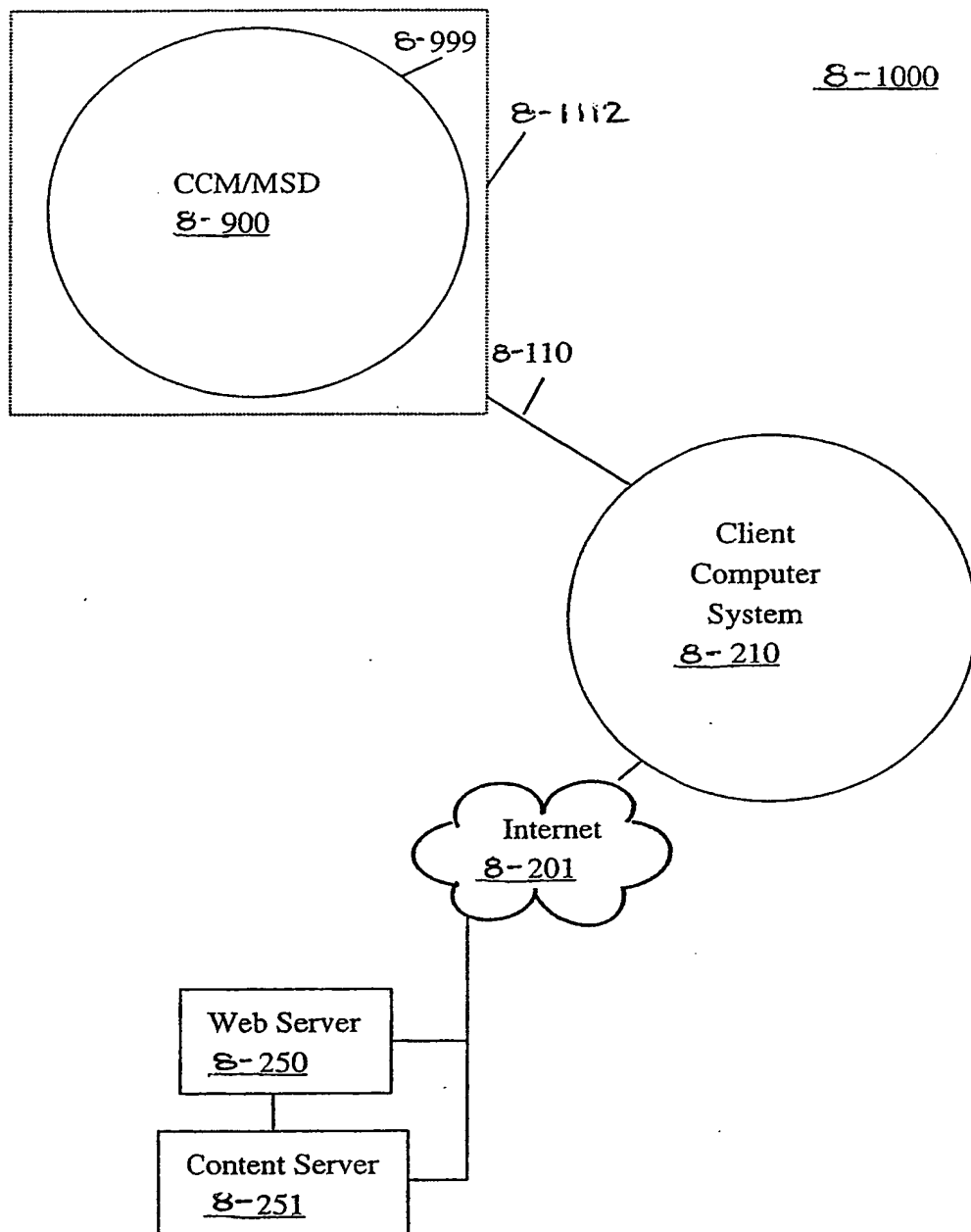


FIGURE 8-10

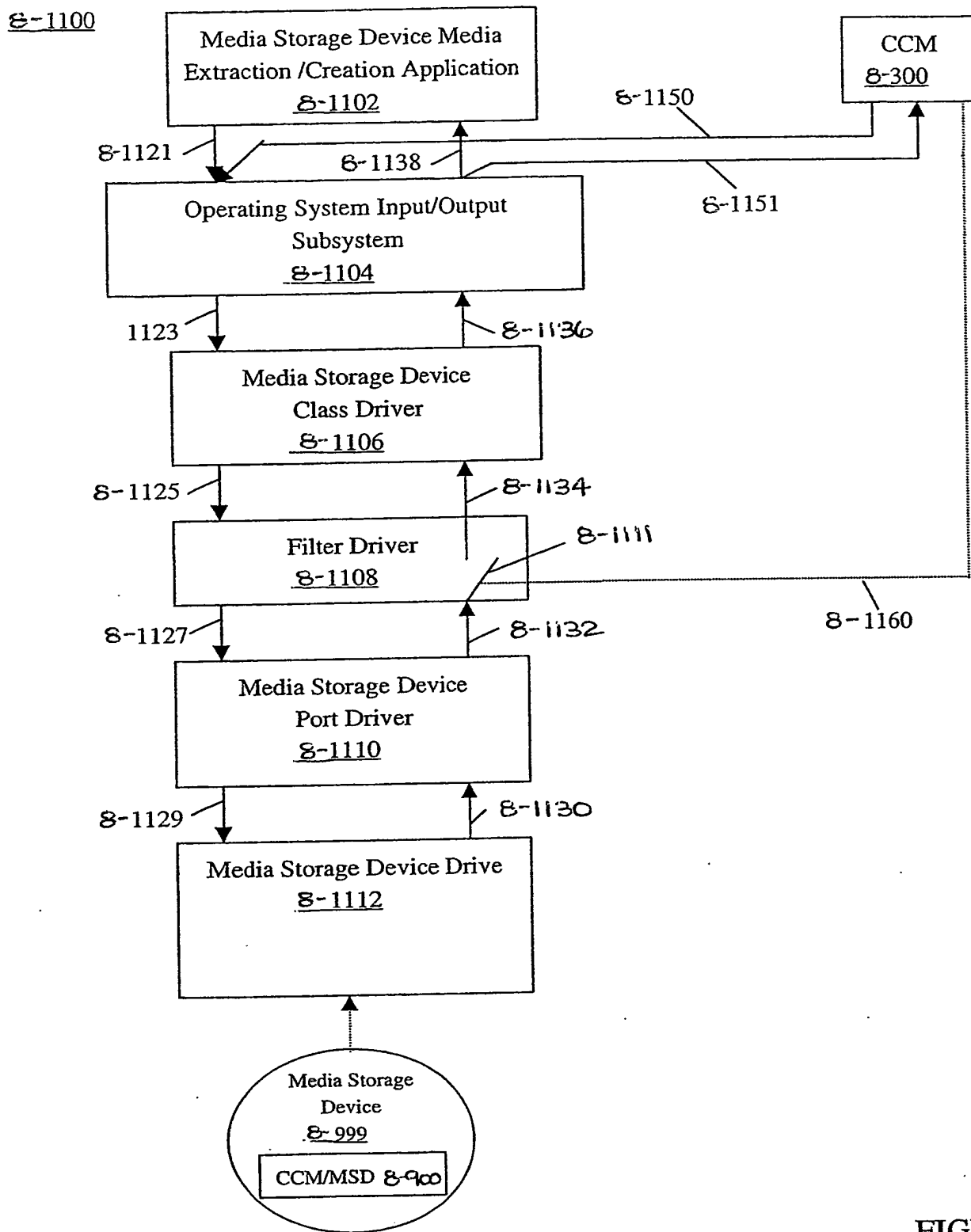


FIGURE 8-11

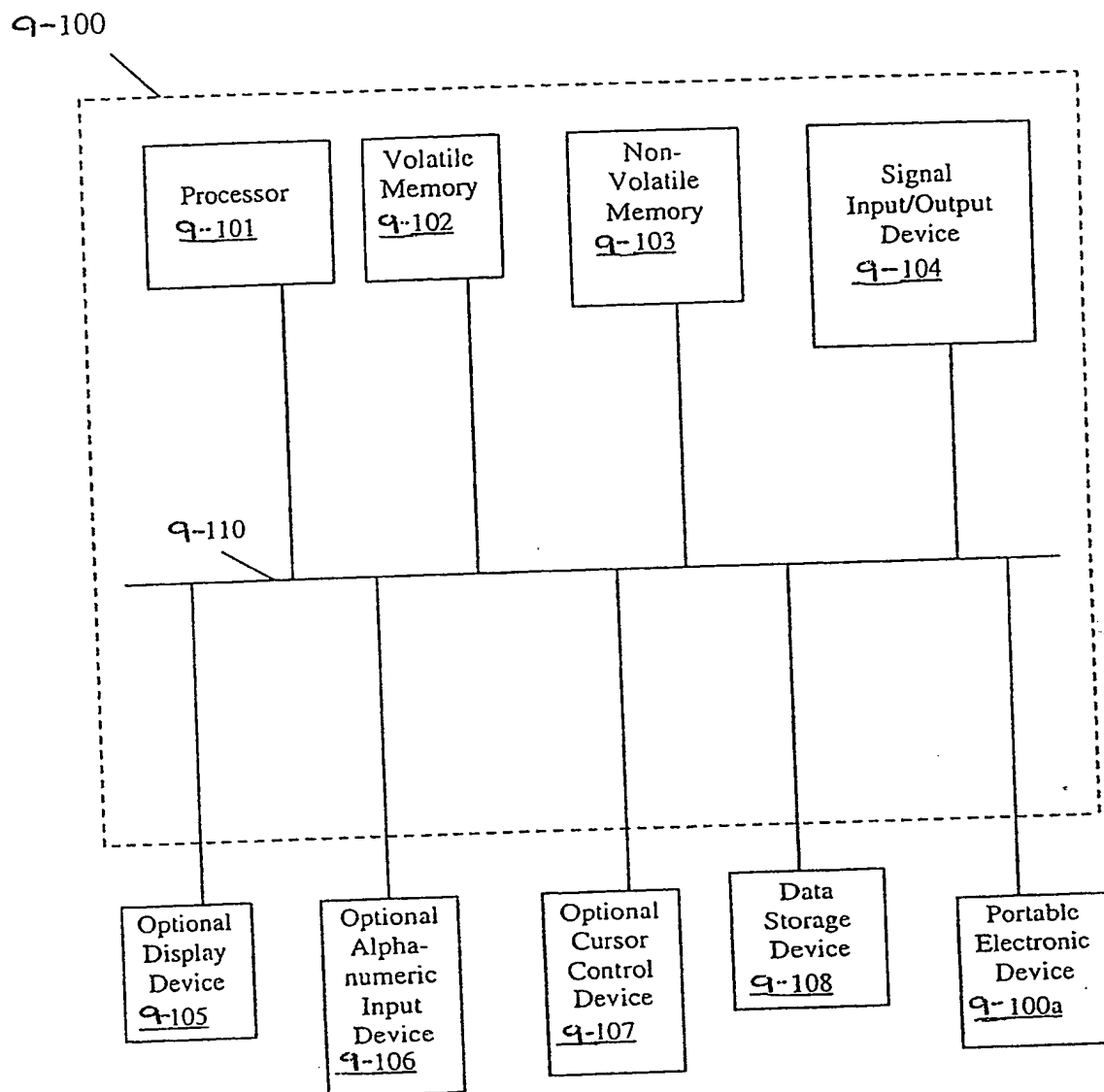


FIGURE 9-1

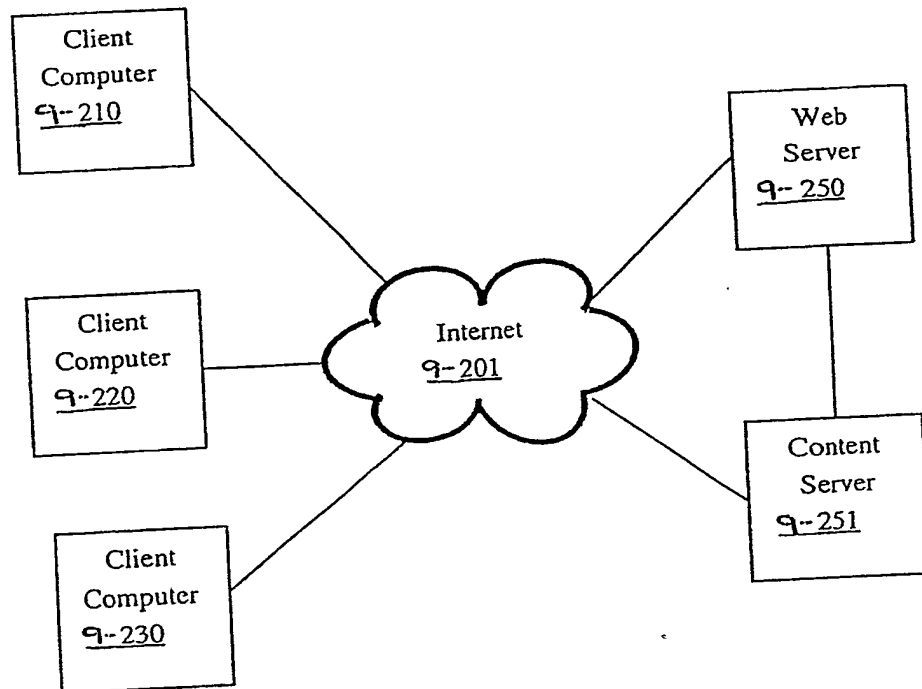
9-200

FIGURE 9-2

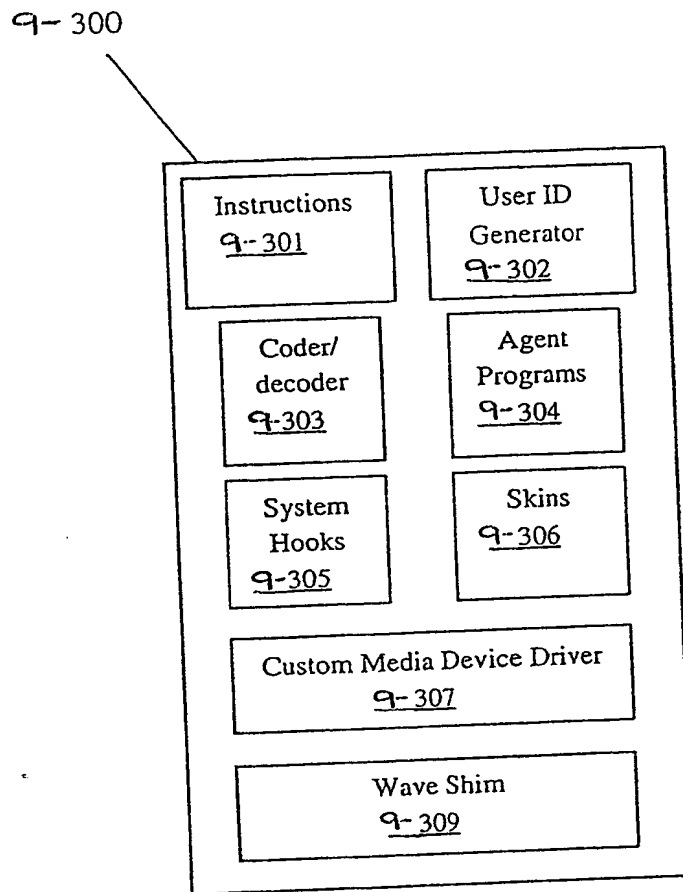


FIGURE 9-3

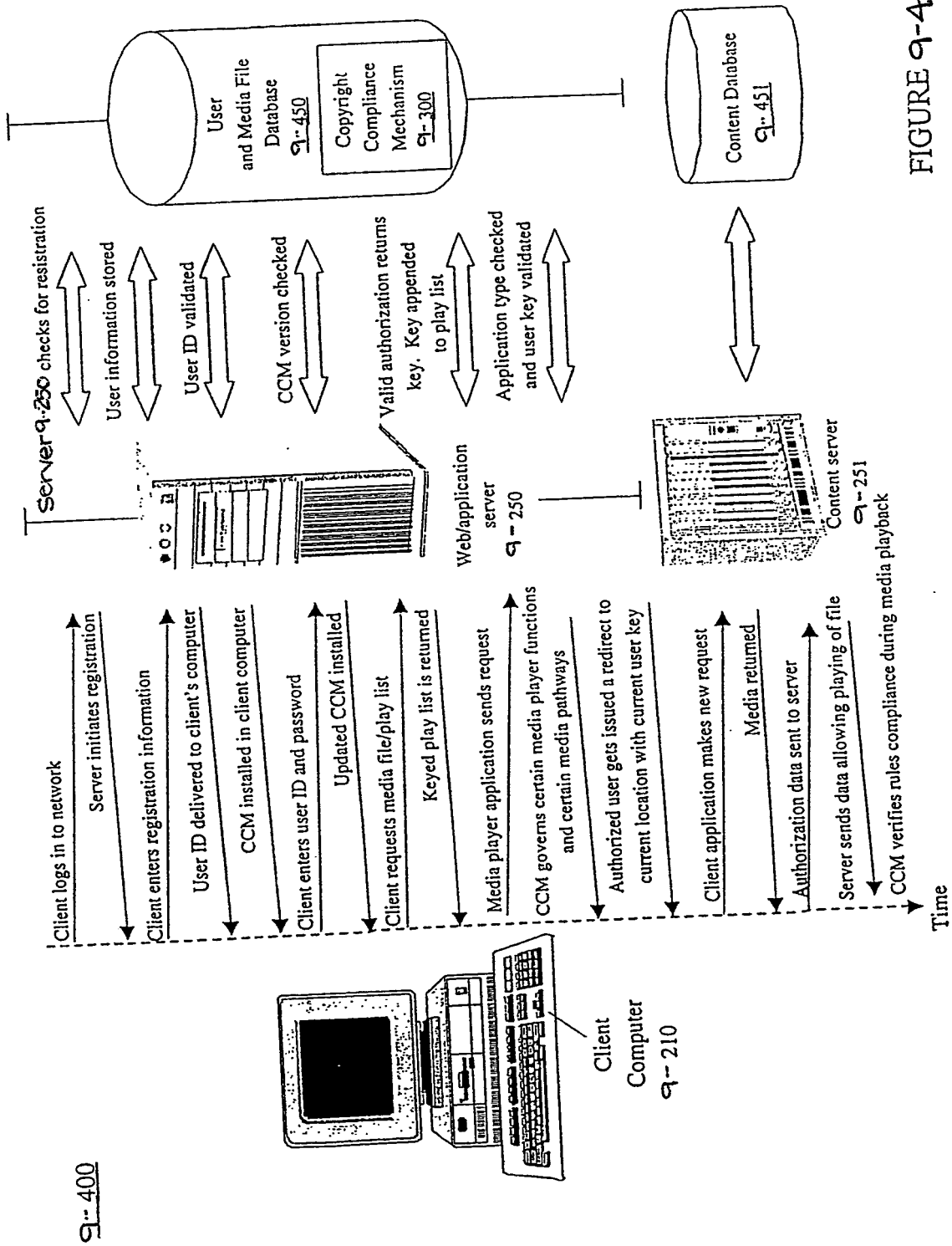


FIGURE 9-4

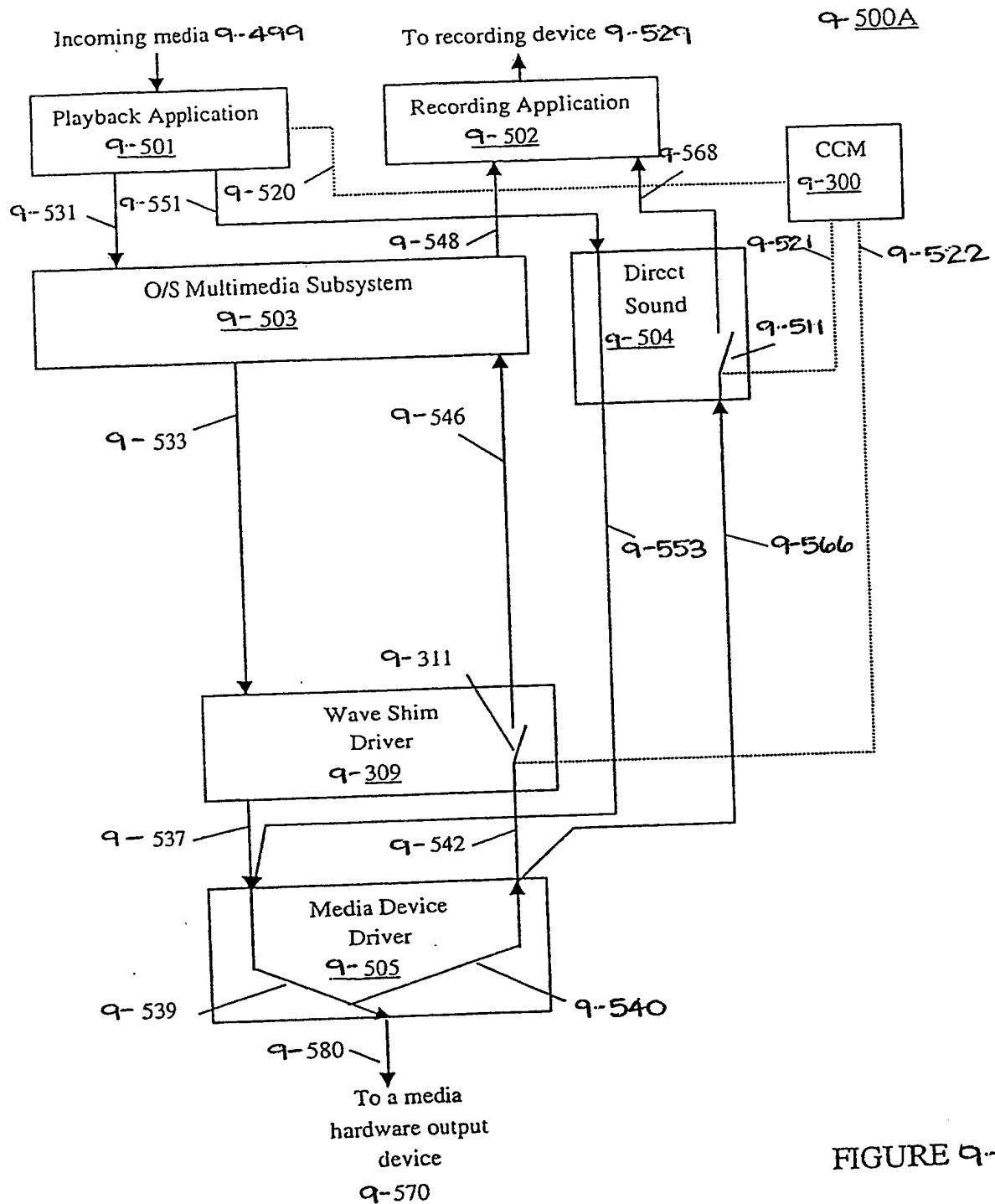


FIGURE 9-5A

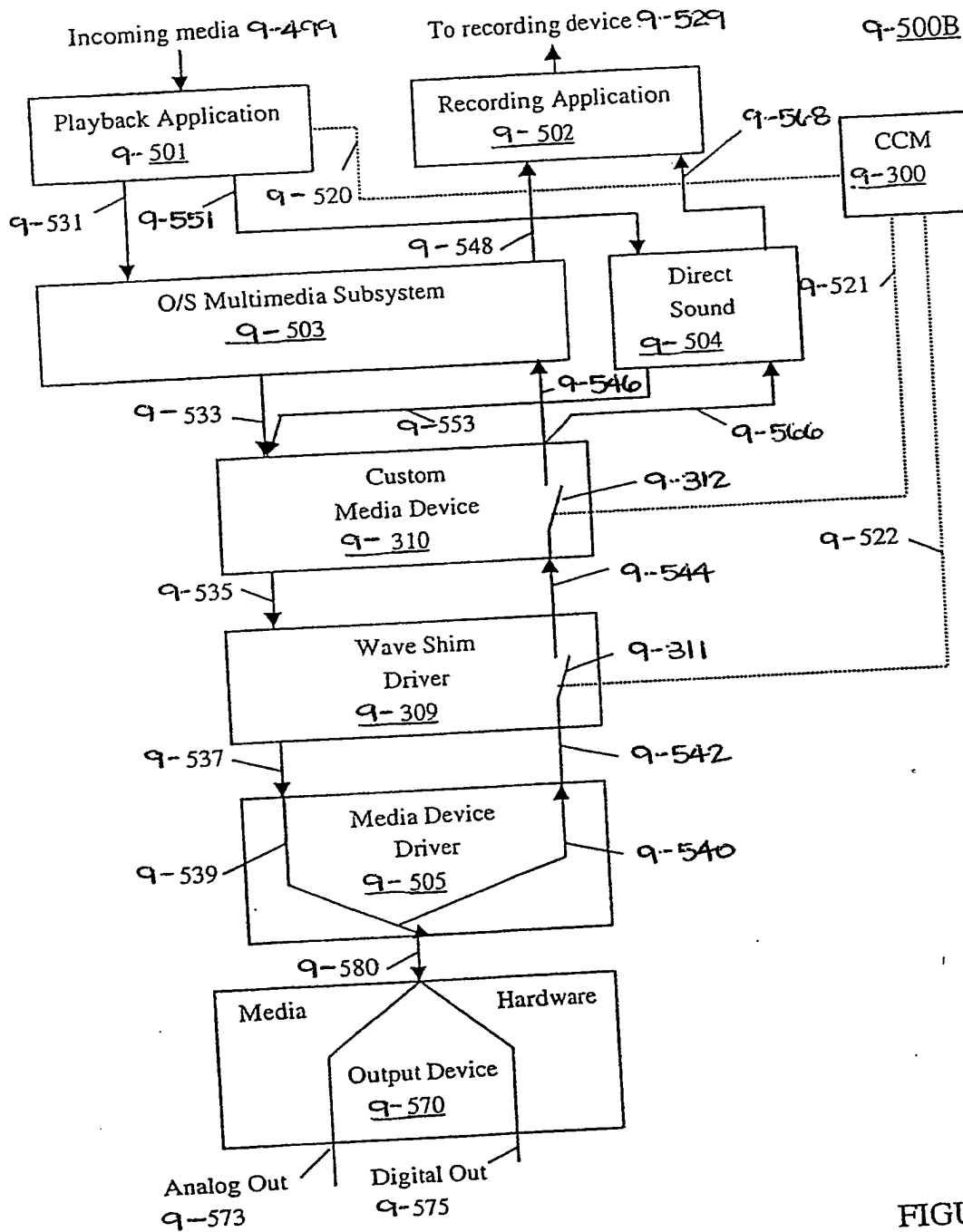


FIGURE 9-5B

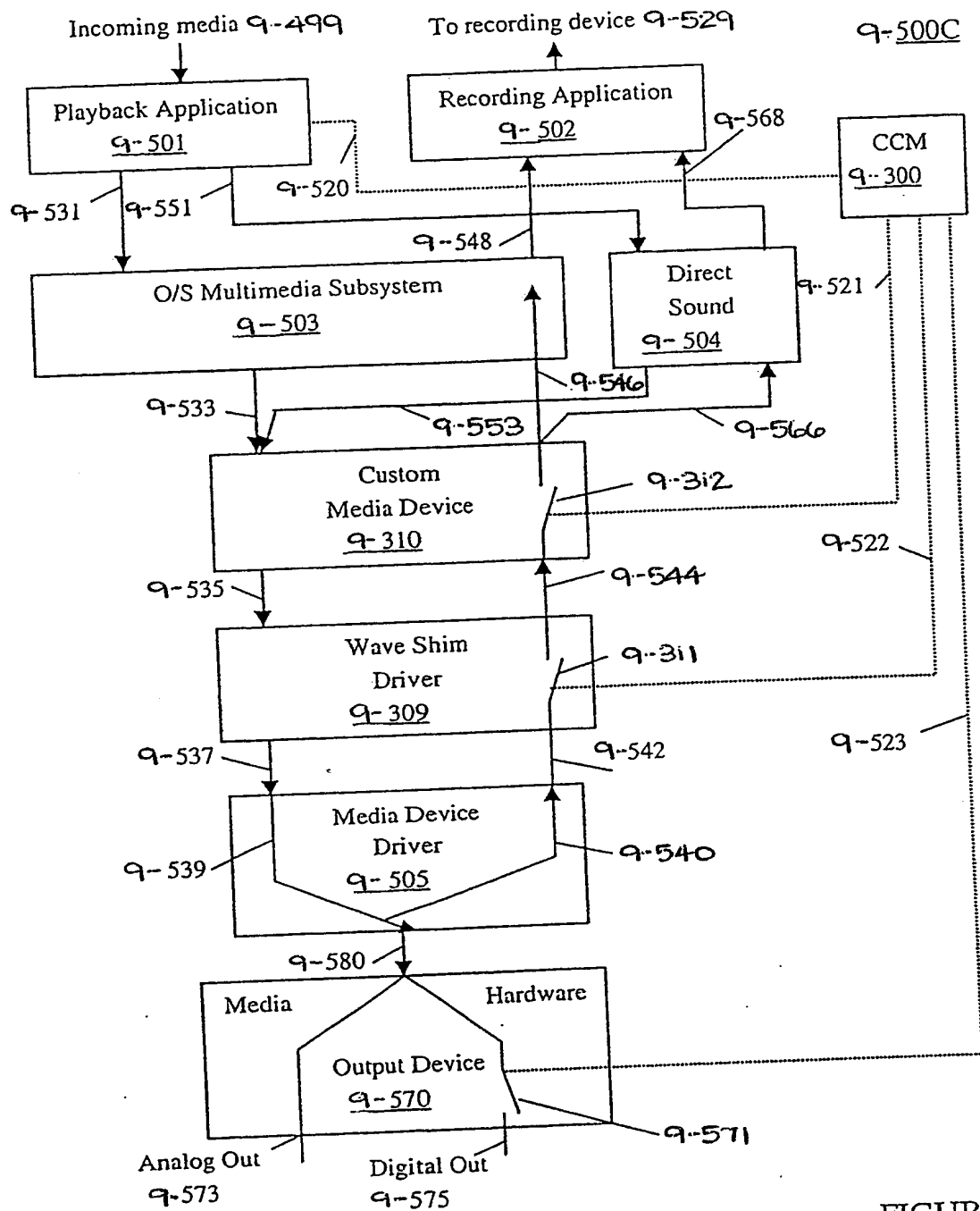


FIGURE 9-5C

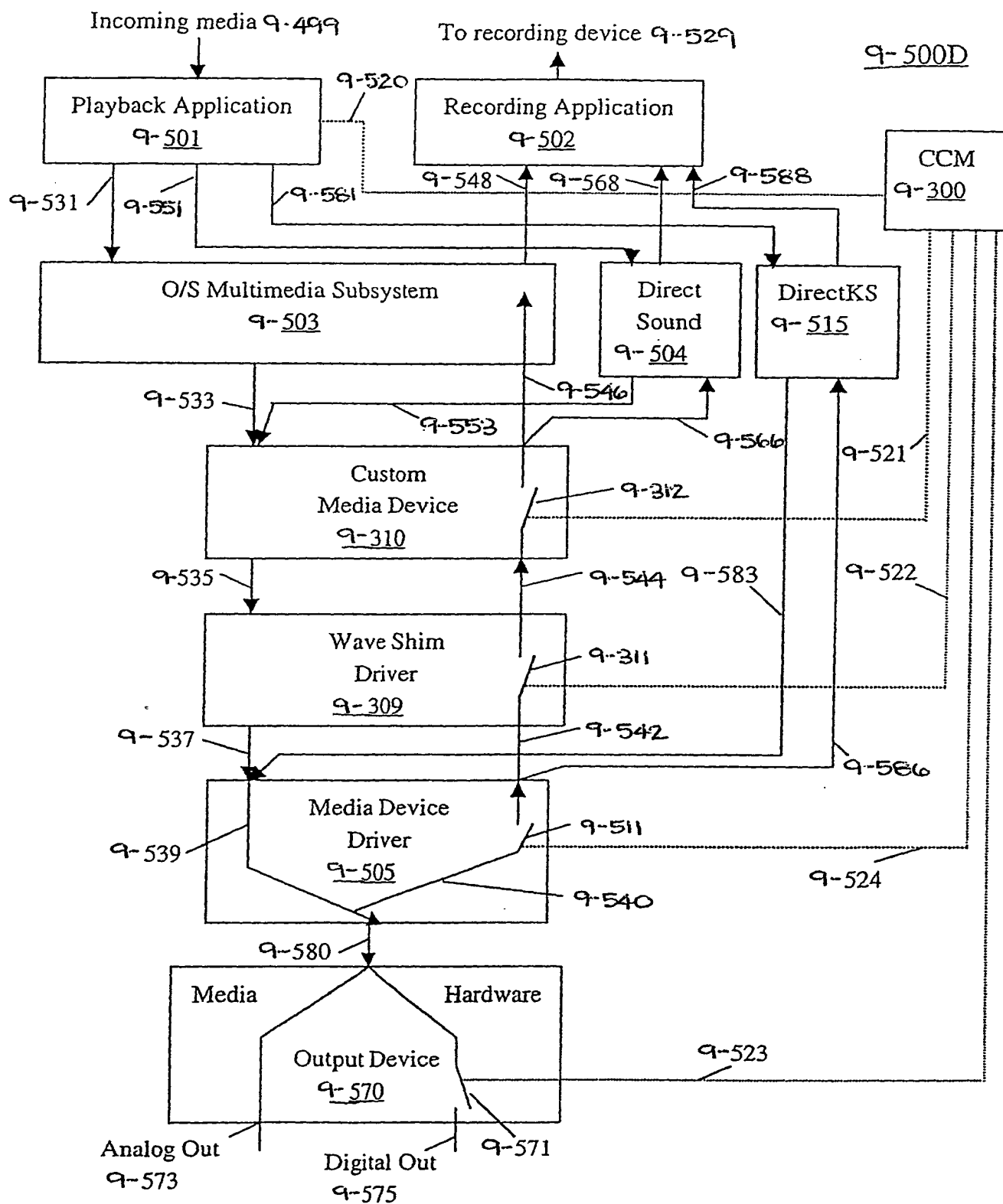


FIGURE 9-50

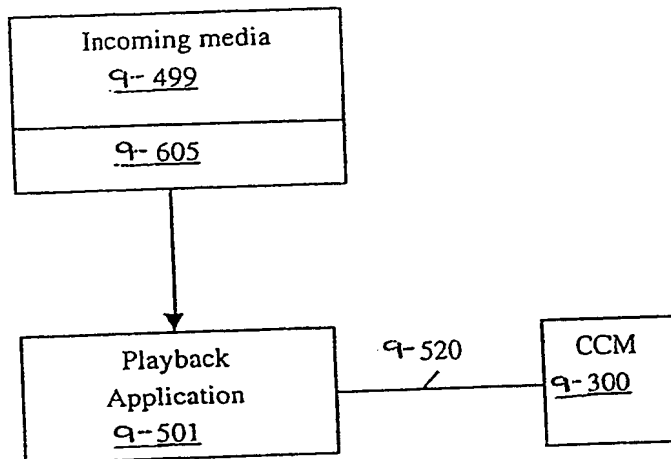
9-600

FIGURE 9-6A

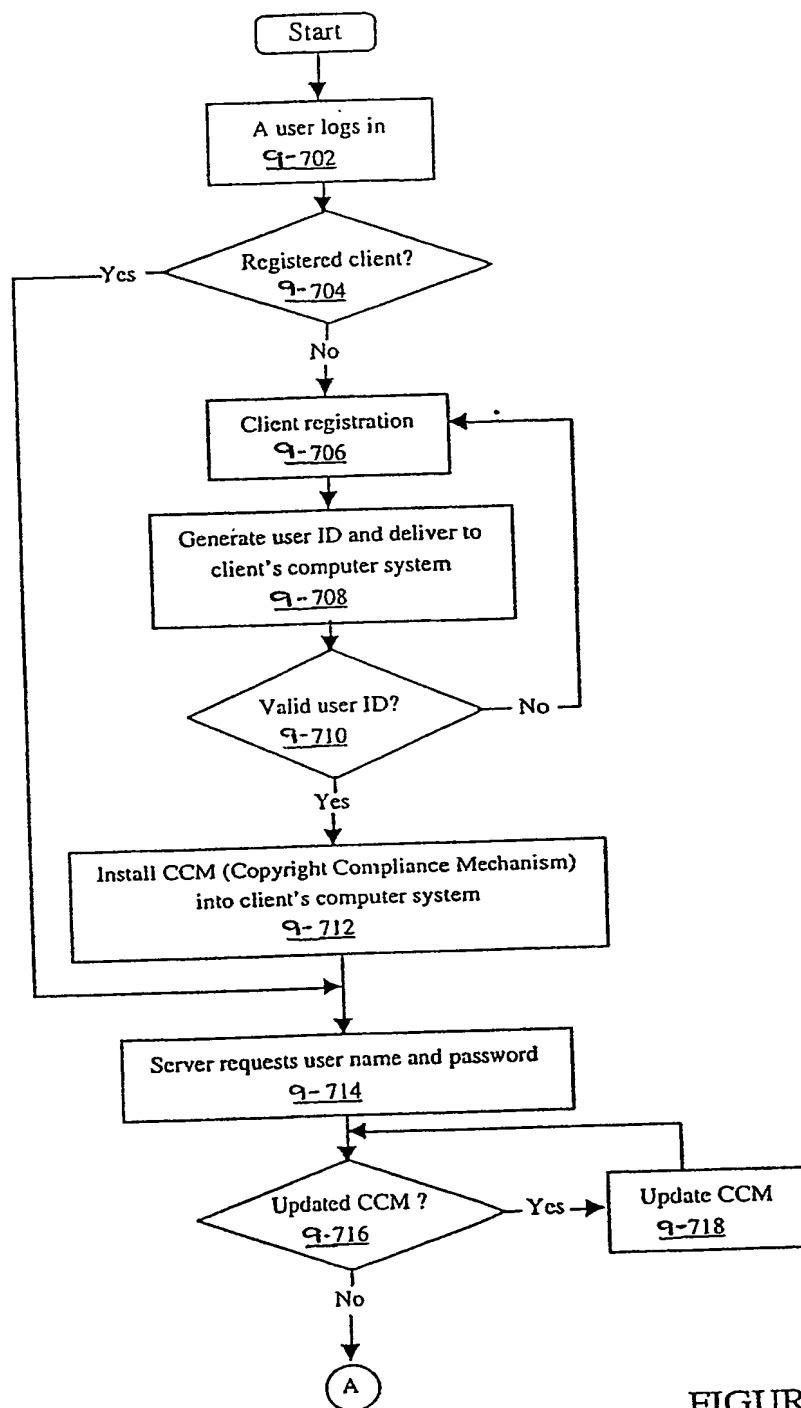
9-700

FIGURE 9-7A

9-700

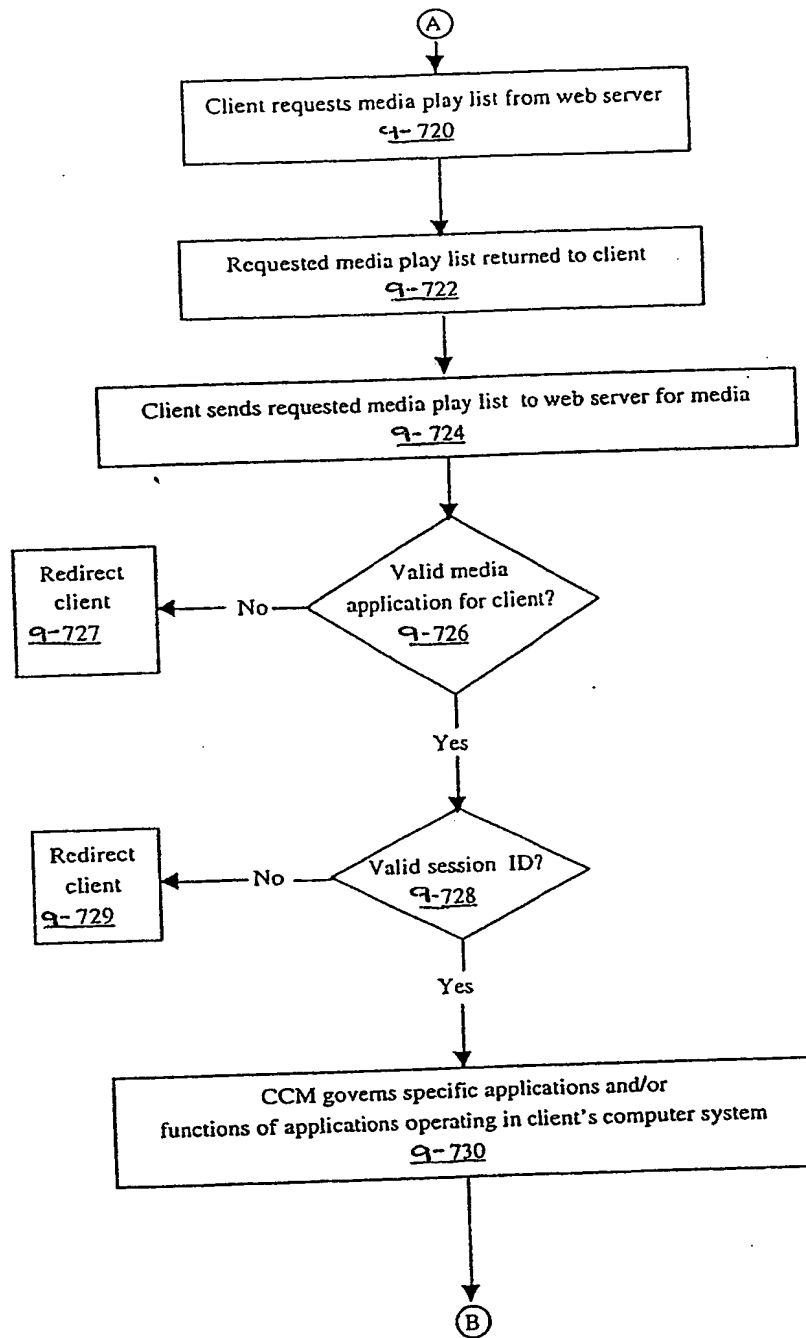


FIGURE 9-7B

9-700

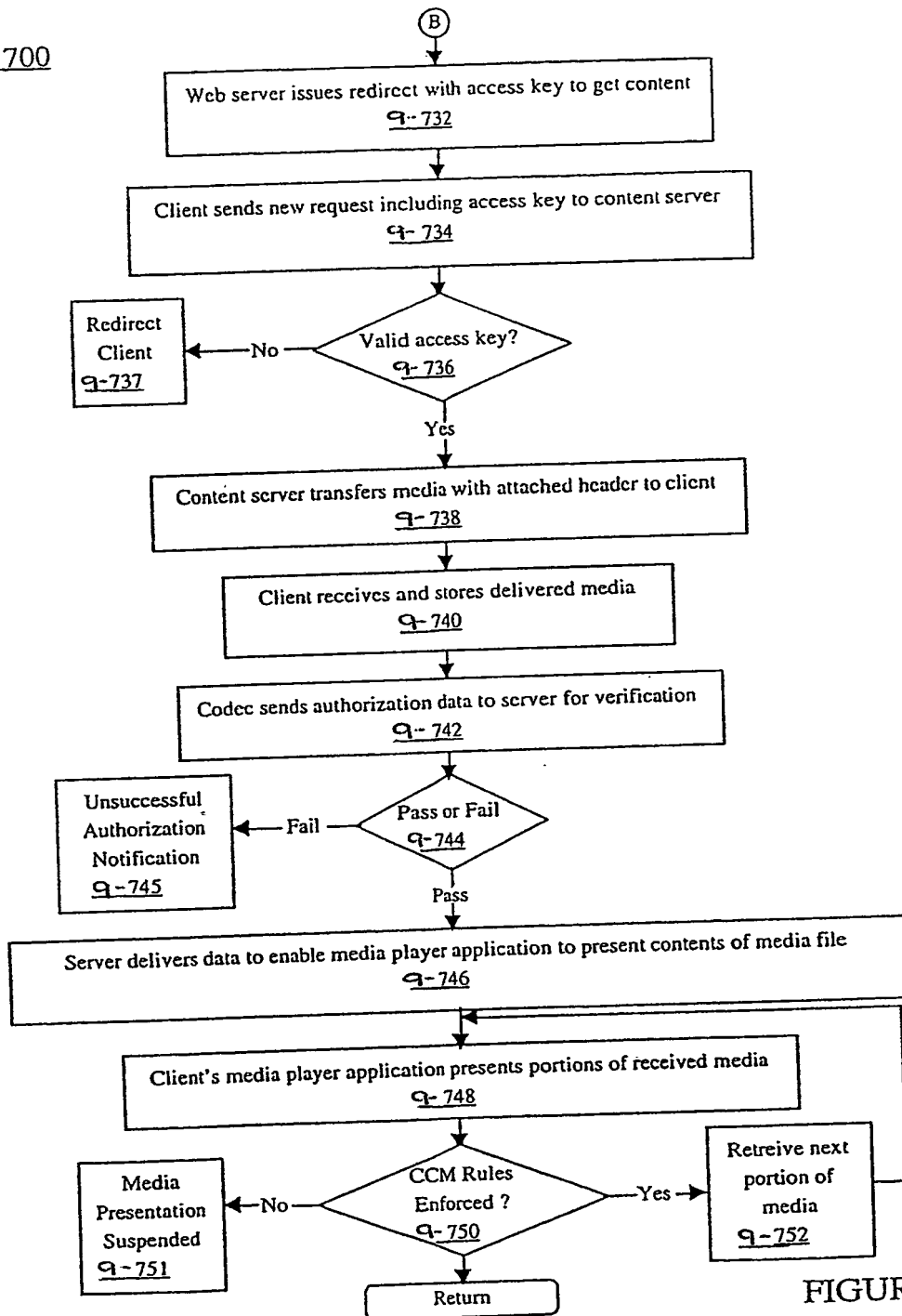


FIGURE 9-7C

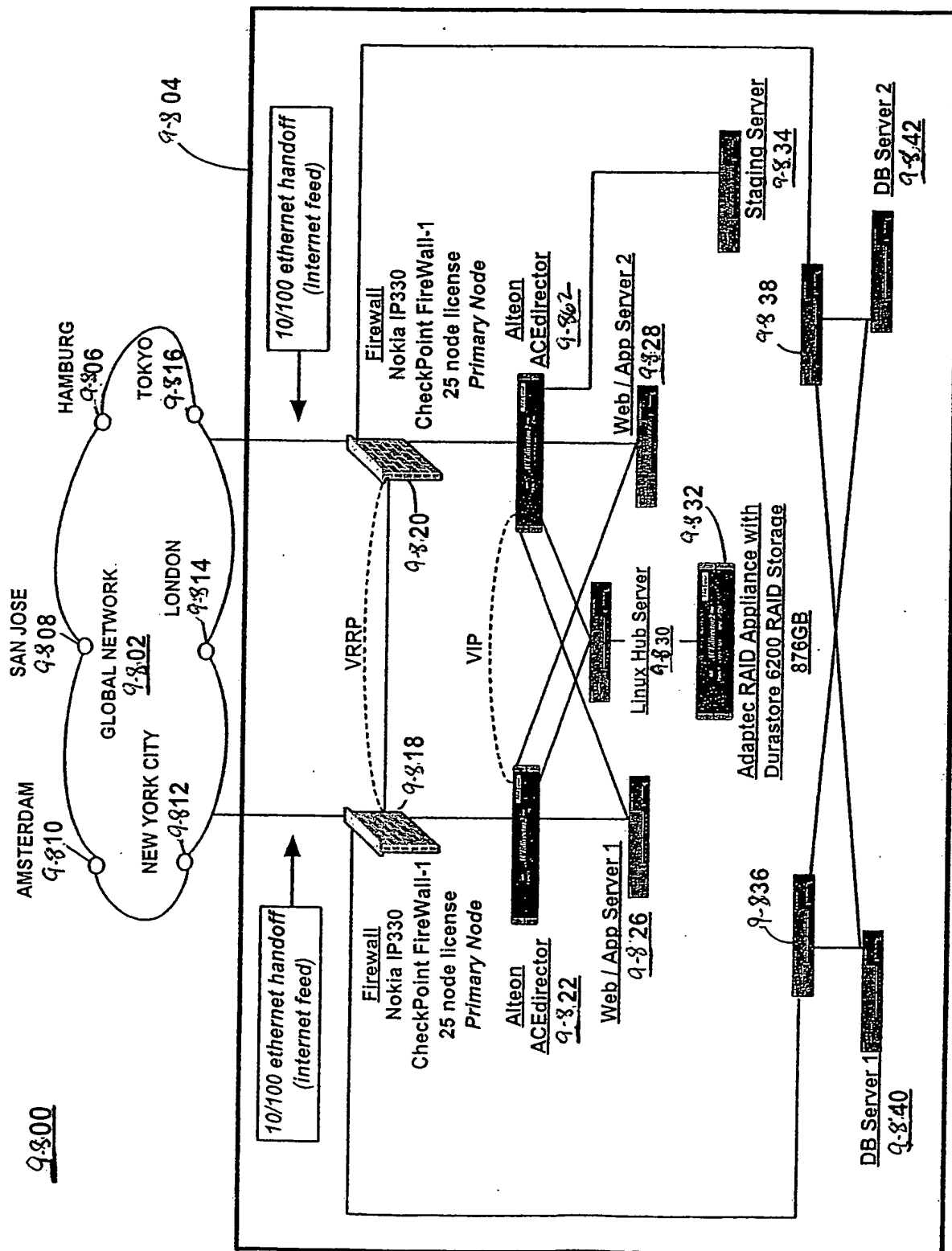


FIG. 9-8

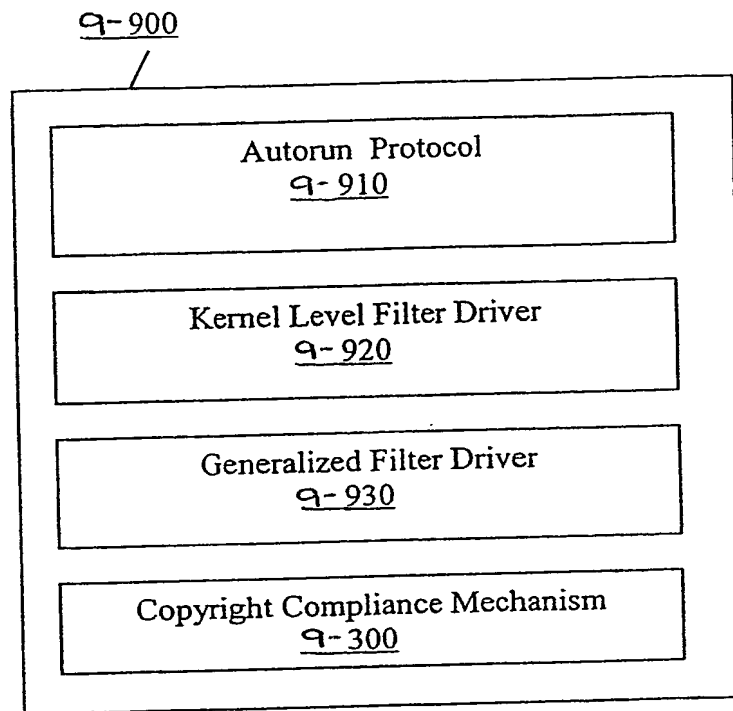


FIGURE 9-9

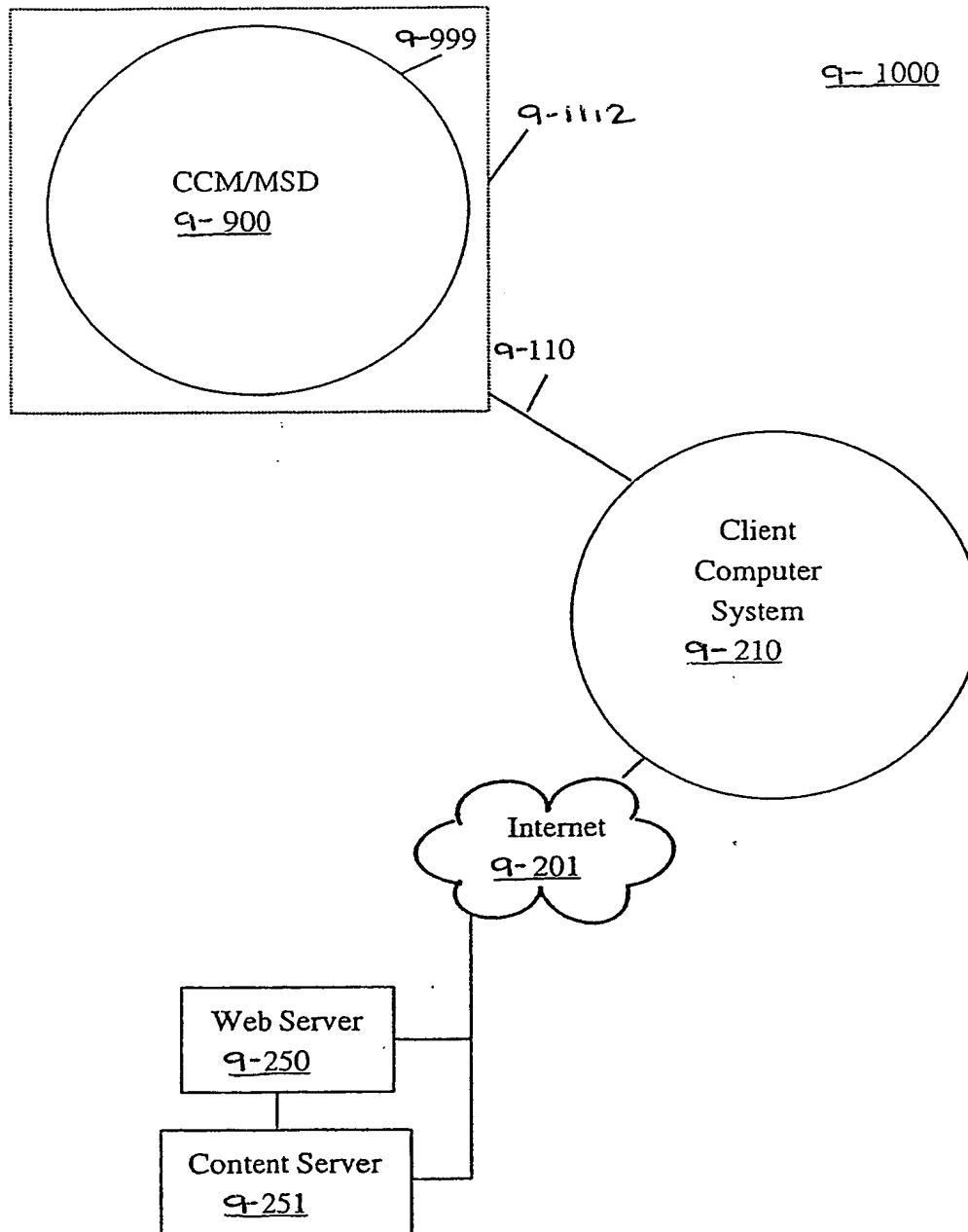


FIGURE 9-10

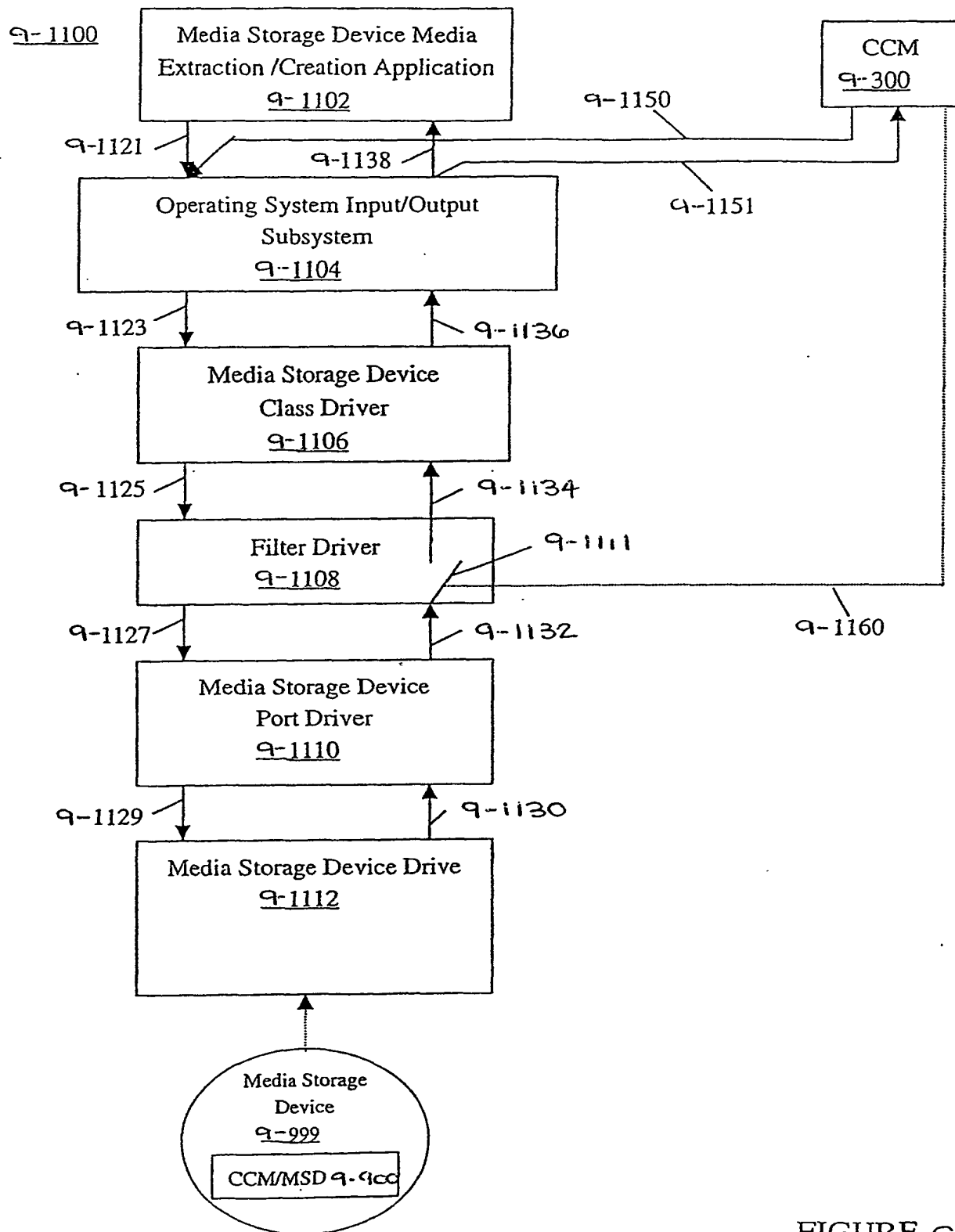


FIGURE 9-11

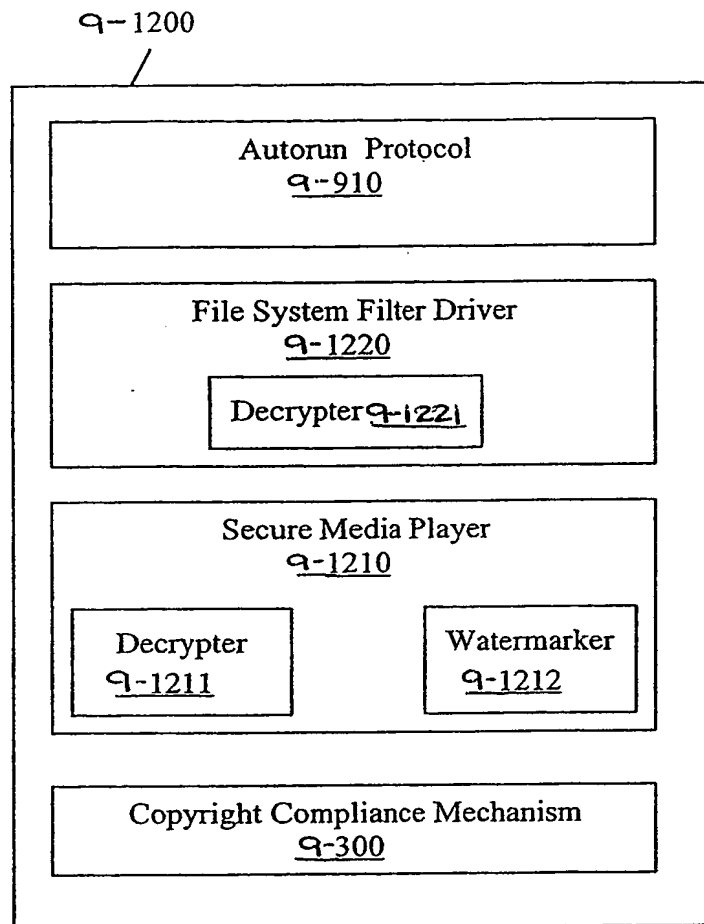


FIGURE 9-12

9-999

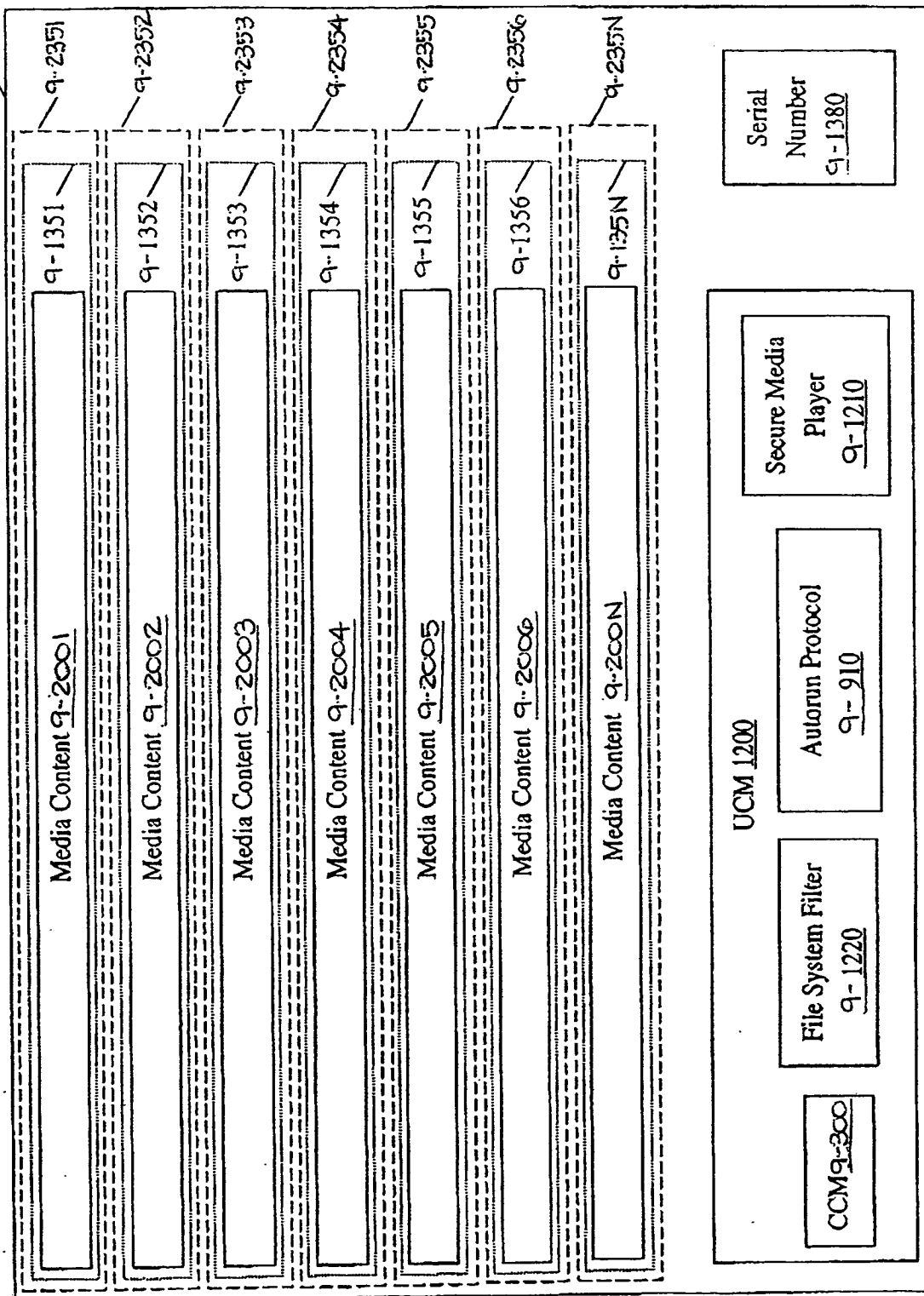


FIGURE 9-13

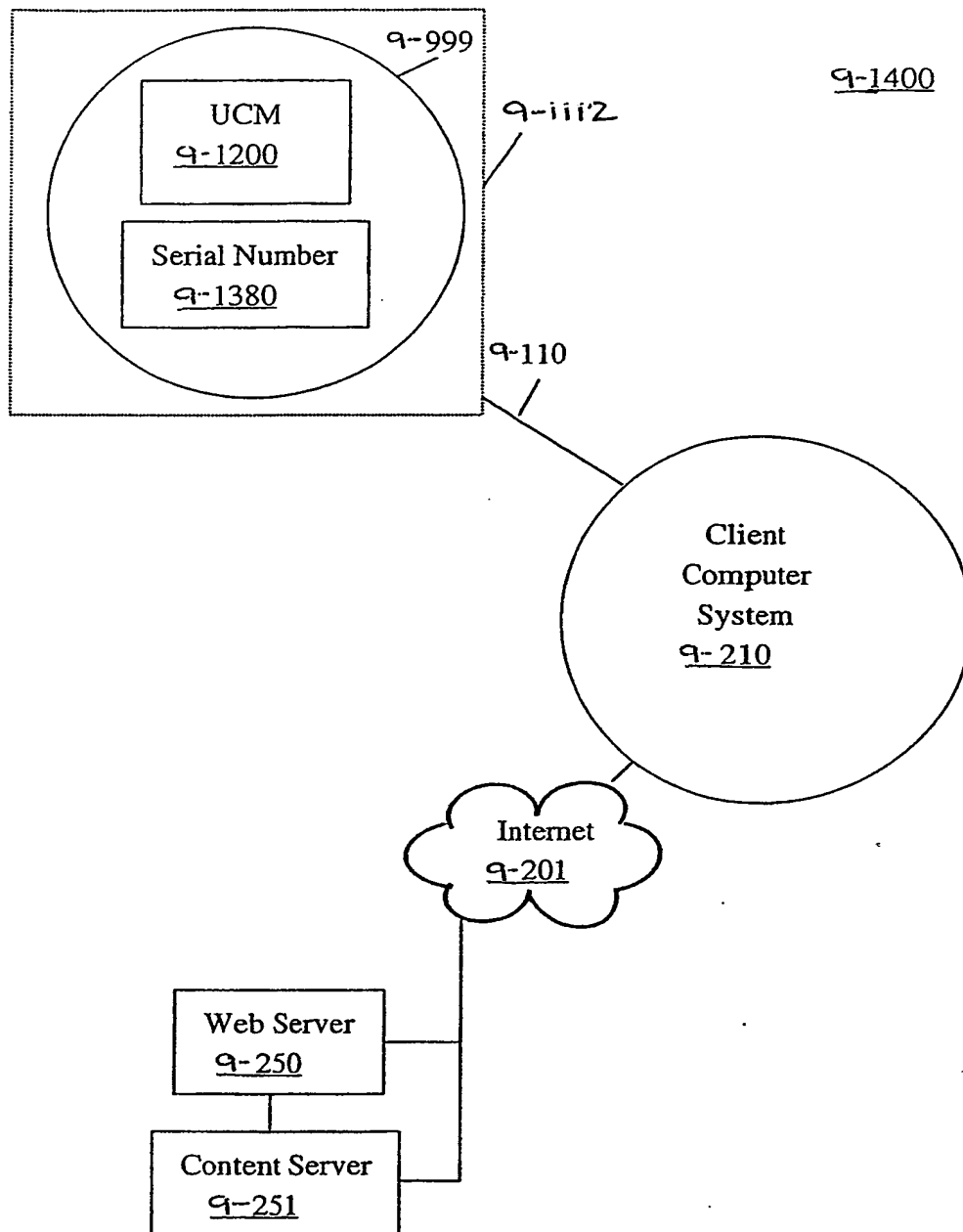


FIGURE 9-14

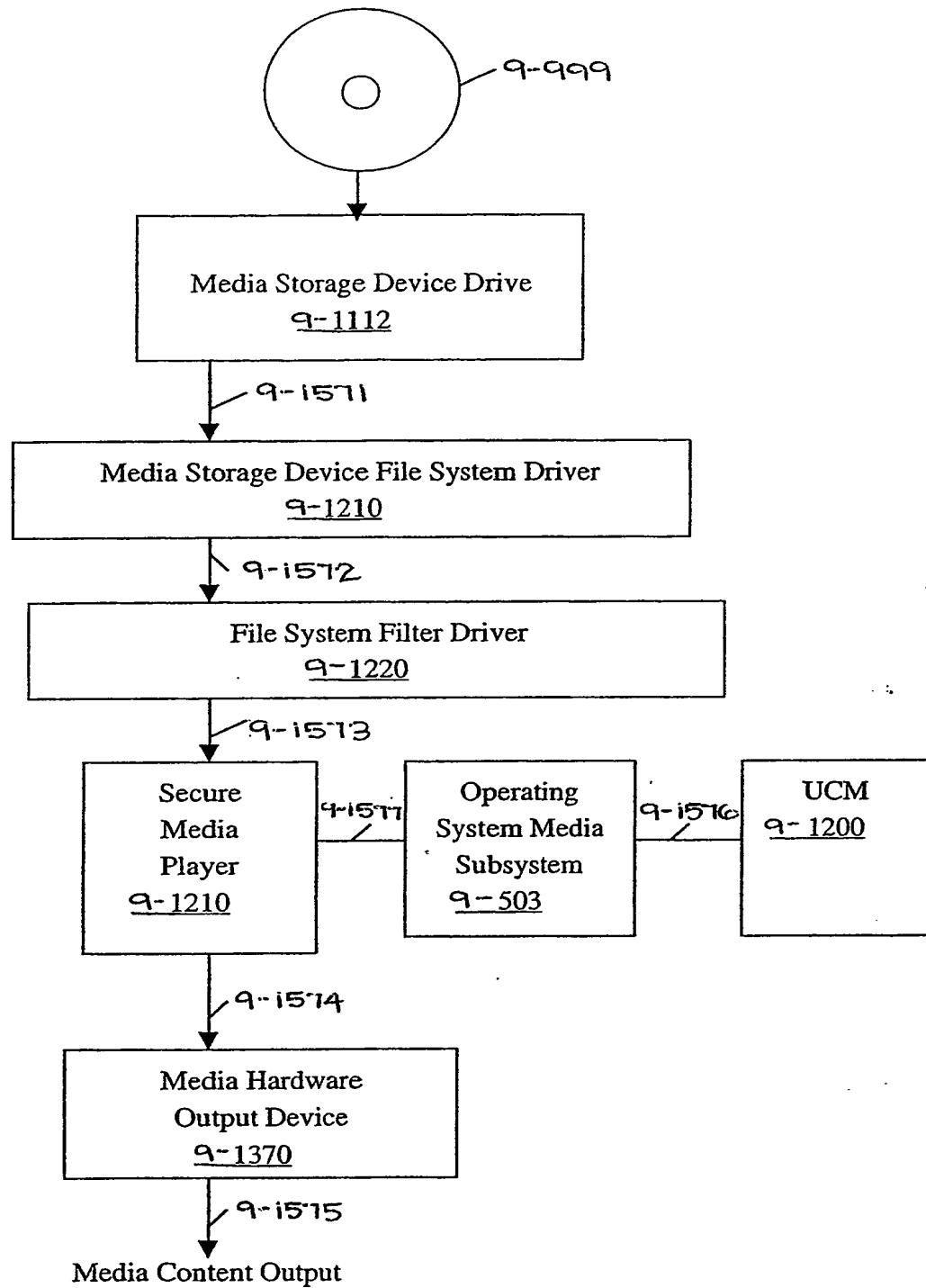
9-1500

FIGURE 9-15

9-1600

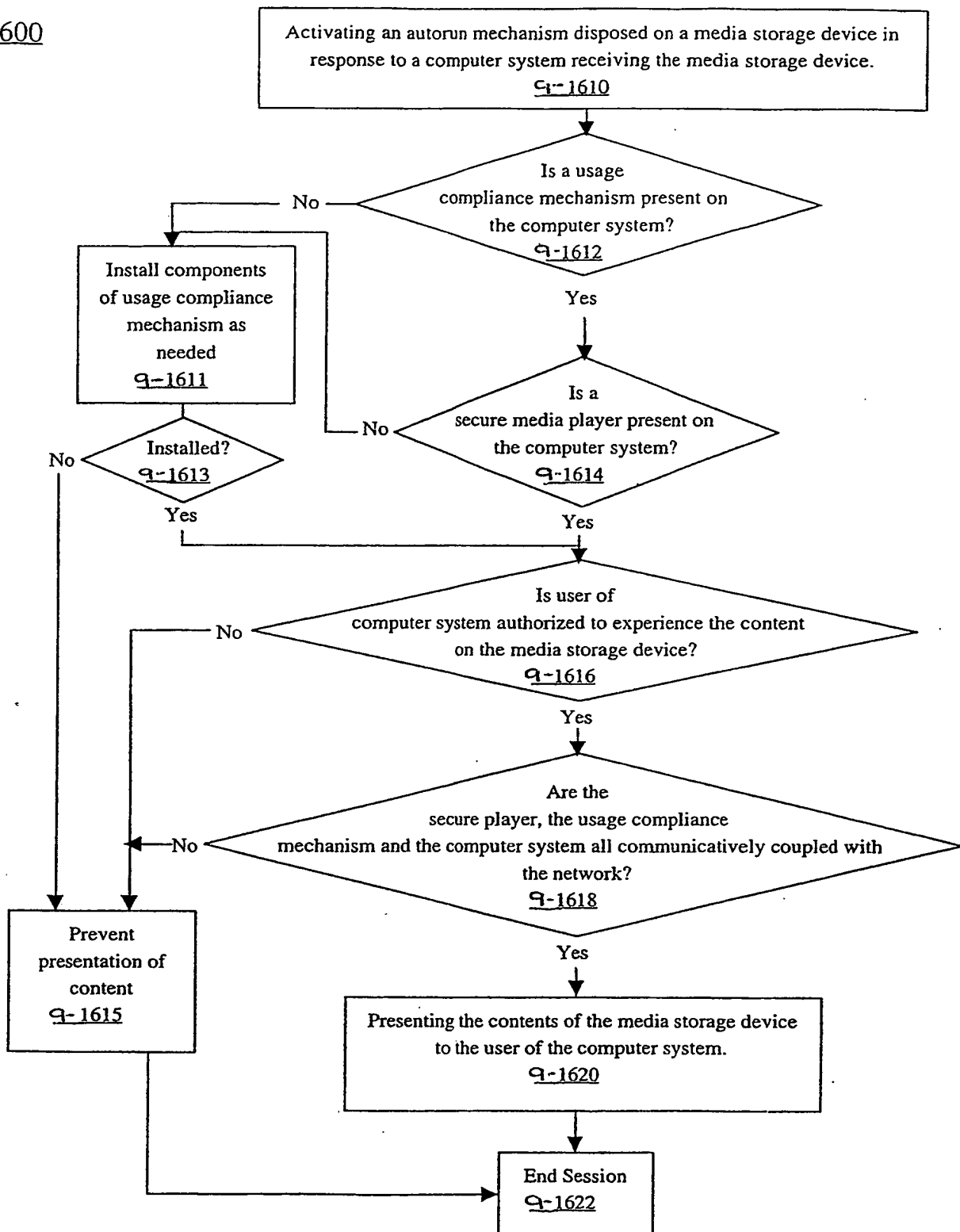


FIGURE 9-16

REVISED VERSION

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
20 November 2003 (20.11.2003)

PCT

(10) International Publication Number
WO 2003/096340 A2

(51) International Patent Classification⁷: **G06F 1/00**

(21) International Application Number:
PCT/US2003/014878

(22) International Filing Date: 10 May 2003 (10.05.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/379,979	10 May 2002 (10.05.2002)	US
60/378,011	10 May 2002 (10.05.2002)	US
10/218,241	13 August 2002 (13.08.2002)	US
10/235,293	4 September 2002 (04.09.2002)	US
10/304,390	25 November 2002 (25.11.2002)	US
10/325,243	18 December 2002 (18.12.2002)	US
10/364,643	10 February 2003 (10.02.2003)	US
60/451,231	28 February 2003 (28.02.2003)	US
10/430,843	5 May 2003 (05.05.2003)	US
10/430,477	5 May 2003 (05.05.2003)	US

(71) Applicants and

(72) Inventors: **RISAN, Hank** [US/US]; 515 Washington Street, Santa Cruz, CA 95060 (US). **FITZGERALD, Edward, Vincent** [US/US]; 100 Peach Terrace, Santa Cruz, CA 95060 (US).

(74) Agents: **GALLENSON, Mavis, S. et al.**; Ladas & Parry, 5670 Wilshire Boulevard, Suite 2100, Los Angeles, CA 90036 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— with declaration under Article 17(2)(a); without abstract; title not checked by the International Searching Authority

(48) Date of publication of this revised version:

4 March 2004

(15) Information about Correction:

see PCT Gazette No. 10/2004 of 4 March 2004, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND SYSTEM FOR MEDIA

(57) Abstract:



WO 2003/096340 A2

PATENT COOPERATION TREATY

PCT

DECLARATION OF NON-ESTABLISHMENT OF INTERNATIONAL SEARCH REPORT

(PCT Article 17(2)(a), Rules 13ter.1(c) and Rule 39)

REC'D 08 JAN 2004

WIPO


PCT

Applicant's or agent's file reference 620919-9	IMPORTANT DECLARATION	Date of mailing(day/month/year) 05/11/2003
International application No. PCT/ US 03/ 14878	International filing date(day/month/year) 10/05/2003	(Earliest) Priority date(day/month/year) 10/05/2002
International Patent Classification (IPC) or both national classification and IPC G06F1/00		
Applicant RISAN, Hank		

This International Searching Authority hereby declares, according to Article 17(2)(a), that **no international search report will be established** on the international application for the reasons indicated below

1. ☐ The subject matter of the international application relates to:
- a. ☐ scientific theories.
 - b. ☐ mathematical theories
 - c. ☐ plant varieties.
 - d. ☐ animal varieties.
 - e. ☐ essentially biological processes for the production of plants and animals, other than microbiological processes and the products of such processes.
 - f. ☐ schemes, rules or methods of doing business.
 - g. ☐ schemes, rules or methods of performing purely mental acts.
 - h. ☐ schemes, rules or methods of playing games.
 - i. ☐ methods for treatment of the human body by surgery or therapy.
 - j. ☐ methods for treatment of the animal body by surgery or therapy.
 - k. ☐ diagnostic methods practised on the human or animal body.
 - l. ☐ mere presentations of information.
 - m. ☐ computer programs for which this International Searching Authority is not equipped to search prior art.
2. ☒ The failure of the following parts of the international application to comply with prescribed requirements prevents a meaningful search from being carried out:
- ☐ the description ☒ the claims ☐ the drawings
3. ☐ The failure of the nucleotide and/or amino acid sequence listing to comply with the standard provided for in Annex C of the Administrative Instructions prevents a meaningful search from being carried out:
- ☐ the written form has not been furnished or does not comply with the standard.
- ☐ the computer readable form has not been furnished or does not comply with the standard.
4. Further comments:

PLEASE SEE "FURTHER INFORMATION CONTINUED FROM PCT/ISA/203"

Name and mailing address of the International Searching Authority  European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Selwa Harris
--	---

Form PCT/ISA/203 (July 1998)

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 203

In view of the large number and also the wording of the claims presently on file, which render it difficult, if not impossible, to determine the matter for which protection is sought, the present application fails to comply with the clarity and/or conciseness requirements of Article 6 PCT (see also Rule 6.1(a) PCT) to such an extent that a meaningful search is impossible. Consequently, no search report can be established for the present application.

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guideline C-VI, 8.5), should the problems which led to the Article 17(2) declaration be overcome.

THIS PAGE BLANK (USPTO)

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

☒ **BLACK BORDERS**

☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**

☒ **FADED TEXT OR DRAWING**

☒ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**

☐ **SKEWED/SLANTED IMAGES**

☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**

☐ **GRAY SCALE DOCUMENTS**

☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**

☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**

☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

THIS PAGE BLANK (USPTO)